Outils d'exploration de corpus et désambiguïsation lexicale automatique

Laurent Audibert

UNIVERSITÉ D'AIX-MARSEILLE I - UNIVERSITÉ DE PROVENCE
U.F.R. Mathématiques, Informatique et Mécanique (M.I.M.)
N° attribué par la bibliothèque /__/__/__/__/__/__/__/__/__/__/

THÈSE

pour obtenir le titre de

Docteur de l'Université de Provence

Discipline : Informatique

École doctorale de Mathématiques et d'Informatique de Marseille (ED184)
Équipe DEscription Linguistique Informatisée sur Corpus (DELIC)

présentée et soutenue publiquement

par

Laurent AUDIBERT

le lundi 15 décembre 2003

Outils d'exploration de corpus et désambiguïsation lexicale automatique

devant le jury composé de :

M.	**Jean Véronis**	Directeur de thèse
Mme.	Pascale Sébillot	Rapporteur
Mme.	Béatrice Daille	Rapporteur
M.	Pierre Zweigenbaum	Président

Remerciements

Je tiens, en premier lieu, à remercier Jean Véronis, mon directeur de thèse, pour m'avoir accueilli au sein de l'équipe DELIC (DEscription Linguistique Informatisée sur Corpus), pour avoir pris sur son temps chaque fois que je sollicitais ses précieux conseils et pour m'avoir apporté les moyens d'arriver au bout de ce long et difficile chemin.

Je remercie particulièrement Jean-Luc Péris dont la précision et la clair-voyance des conseils en algorithmique, et notamment en C++*, ne connaissent pas d'égal. Sa disponibilité et son amitié furent une aide précieuse, je lui en serai toujours reconnaissant.*

Je tiens à préciser que cette thèse n'aurait pu voir le jour sans l'immense travail de Delphine Reymond pour la constitution du dictionnaire distributionnel et l'étiquetage lexical manuel du corpus SyntSem*.*

Je remercie et suis redevable envers tous mes relecteurs avec, en première ligne, mon frère et ma mère, Sébastien et Danielle Audibert, mais aussi Christophe Rey, Christophe Benzitoun, Delphine Reymond et Sandrine Henry. Mon orthographe quantique ne leur a pas rendu la tâche facile. Je remercie à cette occasion Pascale Sébillot, Sid-Ahmed Berrani, Vincent Claveau et Mathias Rossignol pour leur relecture et leurs commentaires avisés concernant les chapitres 5 et 6.

Je remercie également tous les membres du CILSH *(Centre Informatique pour les Lettres et Sciences Humaines), des administrateurs efficaces et entièrement dévoués à leur mission, des secrétaires à l'identité fluctuante mais à la gentillesse constante, aux enseignants, doctorants, étudiants et autres personnels.*

Je remercie aussi Pascale Sébillot et Béatrice Daille qui m'ont fait l'honneur de rapporter ma thèse ainsi que Pierre Zweigenbaum président du jury.

Je remercie enfin toute ma famille pour son amour et son soutien.

À ma femme, Adina, qui illumine chacun de mes jours.

Éléments de signalement

Titre

« Outils d'exploration de corpus et désambiguïsation lexicale automatique »

Discipline

Informatique

Intitulé et adresse de l'équipe de recherche

Équipe DELIC (DEscription Linguistique Informatisée sur Corpus)
Université de Provence
29 Avenue Robert Schuman
13621 Aix-en-Provence Cedex 1
France

Résumé

Ce travail de thèse adresse le problème de la désambiguïsation lexicale automatique à l'aide de méthodes d'apprentissage supervisé. Dans une première partie, nous proposons un ensemble de puissants outils de manipulation de corpus linguistiques étiquetés. Pour réaliser ces outils, nous avons développé une bibliothèque `C++` qui implémente un langage élaboré et expressif d'interrogation de corpus, basé sur des *méta-expressions régulières*. Dans une seconde partie, nous comparons divers algorithmes d'apprentissage supervisé, que nous utilisons ensuite pour mener à bien une étude systématique et approfondie de différents critères de désambiguïsation, basés sur la cooccurrence de mots et plus généralement de n-grammes. Nos résultats vont parfois à l'encontre de certaines pratiques dans le domaine. Par exemple, nous montrons que la suppression des mots grammaticaux dégrade les performances et que les bigrammes permettent d'obtenir de meilleurs résultats que les unigrammes.

Mots clefs

Désambiguïsation lexicale automatique ; traitement automatique des langues ; concordancier ; analyseur ; expression régulière ; corpus lexicalement étiqueté ; apprentissage supervisé ; cooccurrences ; n-grammes.

Title

« Corpus exploration tools and automatic word sense disambiguation »

Abstract

This thesis deals with automatic word sense disambiguation using supervised learning methods. In the first part, we present a set of powerful tools for processing tagged linguistic corpora. To produce these tools, we developed a C++ library that implements an expressive and elaborate corpus-query language, based on *meta-regular expressions*. In the second part, we compare various supervised learning algorithms. We then use them to perform a systematic and in-depth study of various disambiguation criteria based on word co-occurrence, and more generally on n-gram co-occurrence. Our results are not always in line with some practices in the field. For example, we show that omitting grammatical words decreases performance and that bigrams yield better results than unigrams.

Keywords

Word sense disambiguation ; natural language processing ; concordancer ; parser, regular expression ; sense tagged corpora ; supervised learning ; cooccurrences ; n-grams.

Table des matières

Chapitre 1

Introduction

1.1 Motivation

La plupart des mots ont plusieurs significations. Pourtant, quand une personne entend une phrase contenant un mot ambigu, elle la comprend généralement (sans même percevoir l'ambiguïté) sur la base d'une signification particulière de ce mot. Tout se passe comme si, à un moment donné du processus humain de compréhension de la langue, la bonne signification du mot était automatiquement sélectionnée en fonction du contexte parmi toutes les significations possibles de ce mot.

Nous pouvons désigner par *désambiguïsation lexicale* cette tâche consistant à choisir la bonne signification d'un mot polysémique dans un contexte donné. La désambiguïsation lexicale (ou WSD pour *Word Sense Disambiguation* en anglais) est un domaine de recherche qui suscite énormément d'intérêt depuis les années 1950. En effet, l'accomplissement de cette tâche est nécessaire ou utile dans la plupart des travaux en traitement automatique des langues naturelles (TALN).

Telle que décrite ci-dessus, la tâche consistant à lever l'ambiguïté lexicale semble bien définie. Nous pourrions penser que cette tâche est accomplie dans notre cerveau par un sous-module du processus de compréhension de la langue. Étant indispensable à la plupart des traitements du processus de compréhension de la langue, ce sous-module serait un des premiers à intervenir. Ainsi, notre problème se résume à modéliser ce sous-module par un algorithme d'étiquetage lexical automatique. En y regardant de plus près, il s'avère que ce problème n'est ni aussi simple ni aussi bien défini qu'il n'y paraît.

Le problème de l'ambiguïté du sens des mots a été décrit comme AI-complet. Cela veut dire que ce problème ne peut être résolu que si tous les problèmes difficiles en intelligence artificielle sont résolus. L'ampleur de la difficulté a été mise en avant par Bar-Hillel (1960) dans son fameux article où il prétend ne pas voir comment le sens du mot *pen* dans les phrases « The box is in the pen » [1] et « The pen is in the box » [2] pourrait être déterminé automatiquement. Cet article a servi de support pour le rapport ALPAC (1966) [3] qui fut considéré comme la cause directe de l'abandon d'un grand nombre de recherches dans le domaine de la traduction automatique.

1. Traduction : La boîte est dans l'enclos.
2. Traduction : Le stylo est dans la boîte.
3. En 1964, l'administration américaine commande un rapport, le rapport ALPAC (*Automatic Language Processind Advisory Committee*), qui établit un constat d'échec sur les recherches en traduction automatique, et va conduire à l'arrêt des financements et à la disparition quasi totale des recherches dans le domaine.

1.2 Applications de l'étiquetage lexical automatique

La désambiguïsation lexicale est très utile voire indispensable dans un grand nombre de domaines de recherche en traitement automatique des langues naturelles [4].

Traduction automatique (MT)

La traduction automatique [5] est le domaine par excellence où il est crucial de lever l'ambiguïté lexicale des mots afin d'aboutir à des traductions correctes. Par exemple, la traduction en anglais du mot français *mèche* est *lock*, *wick*, *fuse* ou bien *drill* suivant qu'il s'agit d'une mèche de cheveux, de bougie, de pétard ou de perceuse.

Les algorithmes de traduction automatique ont donné lieu à des applications commerciales. Cependant, entre le début de la conception de telles applications et leur commercialisation, il s'écoule plusieurs années. Il n'existe donc pas d'application en traduction automatique qui tire parti de travaux récents en WSD.

recherche d'informations (IR)

Lever l'ambiguïté des mots d'une requête peut permettre d'affiner la recherche. Par exemple, si nous cherchons des textes traitant des rayons laser, il faut ignorer les textes traitant des rayons de soleil ou encore des rayons de bicyclette.

Les travaux récents en WSD s'inspirent parfois des travaux en IR [6]. En effet, définir si un sens particulier s'applique à l'instance d'un mot est, dans une certaine mesure, analogue à savoir si un document donné est une réponse pertinente à la requête formulée.

Actuellement, les travaux en IR utilisent rarement les techniques d'étiquetage morphosyntaxique ou d'étiquetage lexical [7]. La raison en est que ces techniques ne sont pas assez rapides, robustes ou portables, et qu'elles n'apportent pas toujours d'amélioration substantielle. D'ailleurs, plusieurs expériences ont montré que l'ambiguïté du sens des mots ne dégrade pas beaucoup les performances des algorithmes d'IR (Krovetz & Croft, 1992 ; Sanderson, 1994). De plus, il se produit une désambiguïsation implicite lorsque plusieurs mots clefs d'une requête concordent avec plusieurs mots dans un document (Resnik & Yarowsky, 1997). En fait, Sanderson montre que lorsque la désambiguïsation lexicale est très précise, elle apporte un plus en recherche d'informations, sinon elle dégrade les performances.

D'une certaine manière, les recherches en IR ont, pour l'instant, plus apporté à celles en WSD que le contraire. En effet, les recherches en IR ont progressé en utilisant des méthodes statistiques sur des documents dont la structure linguistique est ignorée. Or, c'est vers ce type d'approches que les recherches en WSD se tournent actuellement.

Lexicographie

Les algorithmes de WSD peuvent être très utiles aux lexicographes. En effet, les lexicographes sont amenés à travailler sur de gros corpus pour observer les emplois des mots. Travailler sur des corpus lexicalement étiquetés leur serait d'une grande aide. Ils pourraient alors faire des recherches sur des emplois de mots utilisés dans un sens spécifique sans être confrontés à un grand nombre de citations inadéquates.

4. (Kilgarriff, 1997b) constitue une bonne introduction sur les utilisations possibles des recherches en WSD.

5. Se référer à (Dagan & Itai, 1994) pour en savoir plus sur le sujet.

6. Gale, Church et Yarowsky (1992a, 1992b) font référence explicitement aux techniques d'IR.

7. Cela n'est pas toujours vrai, (Schütze & Pedersen, 1995) est un exemple d'application de WSD pour de l'IR.

Traitement de la parole

La phonétisation correcte des mots en synthèse de la parole demande une tâche de désambiguïsation. Cette tâche est également utilisée en reconnaissance de la parole pour la segmentation des mots et pour la discrimination d'homophones.

Reconnaissance de caractères (OCR)

Les algorithmes d'OCR ont parfois du mal à faire la différence entre deux mots dont la représentation graphique est très proche comme *môme* et *même*. Les techniques utilisées en WSD peuvent être utiles à la résolution de ce type d'ambiguïté.

Compréhension des langues naturelles (NLU)

Actuellement, les applications de compréhension des langues naturelles sont très spécifiques. Elles sont utilisées dans des domaines bien définis et possèdent toujours une base de connaissances liée à ce domaine. Les applications de WSD n'ont donc pas encore leur place ici. En effet, dans la majorité des cas, les mots importants ne sont pas ambigus dans le domaine particulier de l'application.

Cependant, quand les applications de compréhension des langues naturelles deviendront plus polyvalentes, moins dépendantes d'un domaine particulier, les recherches en WSD leur deviendront indispensables.

Restauration de l'accentuation

Yarowsky (1994b, 1994a) développe des algorithmes qui permettent de restaurer les accents sur des textes ayant perdu toute accentuation, de corriger des fautes d'accentuation dans le cadre des logiciels de correction orthographique et grammaticale et de s'affranchir de la saisie des accents automatiquement ajoutés lors de la frappe du texte (cette technique permet notamment l'utilisation des claviers américains dépourvus de touches d'accentuation).

Cette tâche est, à plus d'un titre, un bon exercice pour un algorithme de WSD :

- le problème est représentatif des difficultés rencontrées dans la résolution des ambiguïtés lexicales ;
- ce type de problème permet de s'affranchir de la difficulté de trouver des textes d'apprentissage étiquetés puisqu'il suffit d'utiliser des textes correctement accentués (très largement disponibles) et de supprimer les accentuations pour générer des corpus de test ;
- ce problème débouche directement sur plusieurs types d'applications pratiques, voire commerciales.

1.3 Positionnement et objectifs de ce travail de thèse

Ce travail de thèse adresse le problème de la désambiguïsation lexicale automatique. Dans ce domaine, l'essentiel des recherches effectuées porte sur la langue anglaise. Bien que les résultats obtenus soient parfois généralisables à plusieurs langues, nous avons choisi de travailler sur la langue française. Notre étude nous a amené à développer nos propres applications, et notamment des outils d'exploration de corpus. Cette phase

préliminaire de développement constitue certainement l'effort le plus important que nous ayons eu à fournir au cours de ce travail [8].

Il existe plusieurs approches pour aborder le problème de la désambiguïsation lexicale automatique. De manière générale, nous pouvons distinguer les approches basées sur la modélisation des connaissances ou du raisonnement, les approches utilisant des bases de connaissances et les approches basées sur corpus. Parmi les approches basées sur corpus, certaines ont besoin de corpus lexicalement désambiguïsés tandis que d'autres s'affranchissent de cette limitation. Il semble aujourd'hui clair que les approches basées sur des corpus lexicalement désambiguïsés sont celles qui obtiennent les meilleurs résultats (Ng, 1997 ; Escudero, Marquez & Rigau, 2000a ; Kilgarrif & Rosenzweig, 2000 ; Agirre & Martinez, 2001a). Ce type d'approche cherche à résoudre le problème en utilisant des techniques de classification supervisée. Un algorithme extrait automatiquement les connaissances nécessaires pour la désambiguïsation à partir d'un grand corpus lexicalement désambiguïsé, dans lequel les mots ont été manuellement étiquetés selon les lexies d'un dictionnaire donné. Cette phase d'extraction automatique des connaissances est appelée apprentissage. À l'issue de cette phase, l'algorithme de désambiguïsation est capable d'assigner la lexie adéquate aux mots apparaissant dans une nouvelle phrase, en se basant sur les connaissances acquises durant la phase d'apprentissage. Comme l'attestent les successives campagnes d'évaluation SENSEVAL, de nombreuses études récentes ont recours à ces techniques d'apprentissage.

Actuellement, l'une des difficultés majeures en désambiguïsation lexicale automatique réside dans l'inadéquation des dictionnaires traditionnels (Véronis, 2001) ou dédiés (Palmer, 1998). Par exemple, en utilisant un dictionnaire dédié, le degré d'accord entre annotateurs peut n'atteindre que 57% (Ng & Lee, 1996) et être équivalent à une affectation aléatoire des lexies en utilisant un dictionnaire classique (Véronis, 1998). Une autre difficulté provient du manque de corpus lexicalement désambiguïsés sur lesquels des méthodes d'apprentissage supervisé peuvent être entraînées. Ce manque se transforme même en absence totale pour une langue comme le français. Pour ces multiples raisons, notre équipe a entrepris la construction d'un dictionnaire distributionnel en se basant sur un ensemble de critères différentiels stricts. Ce dictionnaire comporte pour l'instant la description détaillée de 20 noms, 20 verbes et 20 adjectifs et a été utilisé pour étiqueter manuellement chacune des 53 796 occurrences de ces 60 vocables dans le corpus du projet SYNTSEM [9] (corpus composé de textes de genres variés comportant 6 468 522 *mots*).

La désambiguïsation lexicale s'effectue toujours en utilisant l'information du contexte du mot à désambiguïser. Cette information peut être enrichie par un certain nombre d'annotations (étiquette morphosyntaxique, lemmatisation, etc.). Dans tous les cas, il n'est pas possible d'utiliser toute l'information disponible car elle est bien trop importante et bruitée. Il faut donc se focaliser sur un certain nombre d'indices. Le choix de ces indices, déterminé par ce que nous appelons des critères de désambiguïsation lexicale, est primordial et constitue un enjeu important en désambiguïsation lexicale (Bruce, Wiebe & Perdersen, 1996 ; Ng & Zelle, 1997 ; Pedersen, 2001). L'objectif de ce travail de thèse n'est pas de produire une méthode de désambiguïsation lexicale automatique prête à l'emploi. L'objectif est de proposer une étude rigoureuse, systématique et approfondie des critères pour la désambiguïsation lexicale automatique supervisée pour

8. La réalisation du dictionnaire distributionnel et l'étiquetage lexical manuel du corpus (ces deux points sont abordés plus loin) constituent également un énorme travail qui fait l'objet d'une autre thèse (Delphine Reymond).

9. Le projet SYNTSEM vise à produire un corpus français d'*amorçage* étiqueté au niveau morphosyntaxique, lemmatisé et comportant un étiquetage syntaxique peu profond (*shallow parsing* en anglais) ainsi qu'un étiquetage lexicale de 60 mots-cibles (cf. section 4.3.1).

le français. Les ressources dont nous disposons nous ont permis d'étudier des critères basés sur la cooccurrence de mots et plus généralement de n-grammes (juxtaposition de un ou plusieurs mots), dans le contexte de 60 mots polysémiques cibles. Nous avons cherché à apporter des éléments de réponse à de nombreuses questions comme la taille et la symétrie des contextes à considérer, l'importance de la lemmatisation, de l'ordre des mots, des mots grammaticaux, de la taille des n-grammes utilisés, etc.

1.4 Description des chapitres

Le prochain chapitre (**chapitre 2**) est consacré à l'état de l'art en désambiguïsation lexicale automatique. Après un bref historique, ce chapitre décrit les travaux réalisés dans le domaine de la désambiguïsation en présentant les différentes approches citées dans la section précédente : les approches basées sur la modélisation des connaissances ou du raisonnement, les approches utilisant des bases de connaissances et les approches basées sur corpus.

Très tôt, dans le cadre de ce travail de thèse, des besoins en outils adaptés se sont fait ressentir. Ces besoins ne se limitaient pas au cadre strict de ce travail, puisqu'ils ont commencé en amont, lors de la constitution et de l'étiquetage du corpus, et s'étendent au niveau de l'enseignement et de la recherche au sein de notre équipe. Dans le **chapitre 3** nous passons en revue les outils existants vers lesquels nous aurions pu nous tourner. Nous présentons ensuite la bibliothèque `C++` que nous avons mise au point pour développer nos propres applications.

Le **chapitre 4** présente la constitution et la préparation du corpus du projet Synt-Sem. Il détaille la phase d'étiquetage lexical manuel ainsi que la finalisation du corpus et donne enfin des informations quantitatives sur le corpus, ainsi que sur les 60 vocables de l'étude. (*i.e.* les 60 vocables décrits par le dictionnaire distributionnel).

Le **chapitre 5** pose les principes de la désambiguïsation lexicale supervisée, décrit les étapes de la mise en œuvre d'un critère et les mesures de la qualité de la classification que nous emploierons au cours de notre étude. Ce chapitre présente également une étude préliminaire dont les objectifs sont non seulement de valider notre approche et nos outils, mais aussi d'énoncer un premier critère sur lequel nous pourrons nous appuyer pour mettre au point différents algorithmes de classification dans le chapitre 6.

Le **chapitre 6** présente la problématique de la classification supervisée et discute du problème des bornes inférieures et supérieures des performances auxquelles il faut s'attendre dans le cadre de la désambiguïsation lexicale. Dans ce chapitre sont ensuite décrites plusieurs techniques d'apprentissage comme l'apprentissage bayesien, l'apprentissage de type liste de décisions basée sur une métrique et l'apprentissage basé sur les instances.

L'objectif du **chapitre 7**, qui constitue le cœur de ce travail de thèse, est de réaliser une étude systématique et approfondie de critères basés sur des cooccurrences ou des n-grammes pour la désambiguïsation lexicale. Avant de mener à bien cette étude, nous recensons les travaux proches du nôtre effectués par d'autres équipes. Le problème de la fiabilité des expériences que nous avons réalisées est abordé à la fin du chapitre.

Le **chapitre 8** clôture ce mémoire. Après avoir rappelé nos objectif, nous y dressons un bilan du travail réalisé, avant d'aborder les différentes perspectives envisageables.

1.5 Contenu des annexes

Les annexes débutent avec le manuel d'utilisation de l'application (Win/DOS)LoX (**annexe A** page 239) et celui de l'application CooLoX (**annexe B** page 257).

Quelques exemples d'entrées du dictionnaire distributionnel utilisé dans le cadre de cette étude sont donnés en **annexe C**.

La fréquence des occurrences dans le corpus de chacune des lexies des 60 vocables est précisée en **annexe D**.

Les informations relatives aux étiquettes morphosyntaxiques sont regroupées en **annexe E**.

Dans ce document, nous avons parfois recours à des abréviations ou des acronymes. Nous donnons la signification de chacun d'eux lors de leur première apparition. Nous avons regroupé **page 335** les significations des abréviations et acronymes utilisés dans ce document pour faciliter leur compréhension lors des emplois ultérieurs.

Ce document est ponctué par l'énoncé de quelques définitions et théorèmes. Leur liste est donnée **page 337**.

Une liste des algorithmes présentés est donnée **page 339**.

De nombreux tableaux et figures viennent illustrer ce document. Nous nous sommes efforcé, pour chacun, de fournir une légende autosuffisante pour comprendre l'information qu'ils véhiculent. Une liste des tableaux est donnée **page 341** et une liste des figures **page 343**.

Les références bibliographiques, **page 347**, clotûrent ce document.

Chapitre 2

Désambiguïsation lexicale automatique : état de l'art

Préambule

La désambiguïsation lexicale automatique étant un thème de recherche important, les publications dans ce domaine sont nombreuses et des états de l'art en anglais existent déjà. Pour rédiger celui-ci, nous sommes partis de l'état de l'art relativement complet de Ide et Véronis (1998) [1], auquel nous avons ajouté des informations contenues dans les thèses de Kilgarriff (1992) et de Luk (1996). Nous avons également incorporé des publications récentes (situées entre 1996 et 2002) dans le domaine. Nous espérons ainsi proposer un état de l'art à jour, relativement complet et en français, du domaine de la désambiguïsation lexicale automatique.

2.1 Introduction

Les approches cherchant à résoudre le problème de l'ambiguïté du sens des mots sont nombreuses et variées. Elles peuvent être classées en plusieurs groupes de la manière qui suit.

Modélisation des connaissances ou du raisonnement : ce type d'approche vise à modéliser la compréhension du langage humain ; nous y trouvons indifféremment des approches symboliques et des approches connexionistes.

Approches utilisant des bases de connaissances : ces approches s'appuient généralement sur des bases de connaissances existantes, comme des lexiques tels que LDOCE ou WordNet, des thésaurus ou d'autres bases de connaissances.

Approches basées sur corpus : ces méthodes sont généralement de type statistique et utilisent de gros corpus de texte ; deux types d'approches sont à distinguer, celles utilisant des corpus étiquetés dans la phase d'apprentissage et celles s'affranchissant de cette limitation.

Approches Mixtes : il existe aussi des approches faisant intervenir simultanément plusieurs de ces techniques.

1. Nous avons inclus pratiquement l'intégralité de l'information de la section 2 (« *Survey of WSD methods* ») de l'état de l'art de Ide et Véronis. Cette information se trouve répartie dans les sections 2.2, 2.3, 2.4 et, dans une moindre mesure, dans la section 2.5 du présent état de l'art.

Dans ce chapitre, après un bref historique (**section 2.2**), nous nous intéressons aux travaux réalisés dans le domaine de la désambiguïsation, en présentant les différentes approches suivant les quatre groupes que nous venons de présenter, c'est-à-dire :

- les approches basées sur la modélisation des connaissances ou du raisonnement (**section 2.3**) ;
- les approches utilisant des bases de connaissances (**section 2.4**) ;
- les approches basées sur corpus (**section 2.5**) ;
- les approches mixtes (**section 2.6**).

Nous terminons ce chapitre en synthétisant les informations importantes dans la section *Synthèse et Conclusions* (**section 2.7**).

2.2 Premiers pas en désambiguïsation lexicale

C'est dans le domaine de la traduction automatique que nous trouvons les premières recherches pour résoudre de manière automatique le problème de l'ambiguïté lexicale des mots. En 1949, dans son Memorandum, Weaver (1949/1955) introduit le besoin de désambiguïsation lexicale dans la traduction par ordinateur. Il expose le problème de la manière suivante : en présentant un mot dénué de tout contexte, il est impossible au lecteur de déterminer quel est son sens, en revanche, en fournissant au lecteur un certain nombre de mots qui se trouvent dans le voisinage (aussi bien à gauche qu'à droite) de ce terme central, il est alors possible pour le lecteur de décider de son sens. La question est de déterminer quel est le nombre minimum de mots voisins requis dont le lecteur a besoin pour obtenir le sens correct de ce mot. Kaplan (1955) observe, lors d'une expérience impliquant sept traducteurs, que présenter les deux mots à gauche et à droite du mot à désambiguïser n'est pas plus significatif que de présenter la phrase entière.

En 1955, Reifler (1955) propose la notion de *coïncidences sémantiques* entre un mot et son contexte, comme facteur principal en désambiguïsation. Le rôle des relations syntaxiques commence alors à être pris en considération. Par exemple, le mot anglais *keep* peut être désambiguïsé suivant que son objet est :

- un gérondif, par exemple dans « *he kept eating* » (il continua de manger) ;
- une phrase adjectivale, par exemple dans « *he kept calm* » (il garda son calme) ;
- une phrase nominative, par exemple dans « *he kept a record* » (il a gardé un enregistrement).

À cette époque, l'objectif de la traduction automatique est modeste et se limite à constituer des dictionnaires spécialisés ou *microglossaires* (Oswald, 1952, 1957 ; Oswald & Lawson, 1953 ; Oettinger, 1955 ; Dostert, 1955 ; Gould, 1957 ; Panov, 1960 ; etc.) dont le contenu ne s'applique qu'à un domaine spécialisé (par exemple les mathématiques), et où à un mot donné ne correspond en général qu'un seul sens.

Pour résoudre les problèmes d'ambiguïtés sémantiques, les besoins en représentation des connaissances se font ressentir très tôt. Ainsi, de nombreuses recherches tentent de créer un langage intermédiaire basé sur des principes logiques et mathématiques. Parmi ces recherches, celles de Richens (1958) et de Masterman (1961) conduisent à la notion de *réseau sémantique* (cf. section 2.3.3). La méthodologie adoptée par Masterman est le précurseur des méthodes basées sur les connaissances plus récentes (cf. section 2.4).

En 1949, Weaver précise déjà que des études statistiques sont nécessaires comme première étape pour la désambiguïsation. Suite à ces considérations, certains auteurs cherchent à établir une approche basée sur l'analyse statistique du langage (Richards, 1953 ; Yngve, 1955 ; Parker-Rhodes, 1958). D'autres tentent d'estimer le degré de polysémie des textes et des dictionnaires (Harper, 1957b, 1957a ; Bel'skaja, 1957 ; Pimsleur,

1957). Pimsleur précise à ce sujet que les mots d'un texte peuvent être désambiguïsés avec une précision de l'ordre de 80% en utilisant simplement le sens le plus fréquent. Cette notion est très proche de l'une des mesures de référence (*baseline* en anglais) à laquelle sont comparées la plupart des méthodes actuelles (cf. section 6.3.1 et 6.3.5).

D'une manière surprenante, les premiers travaux menés en désambiguïsation sémantique ont rapidement soulevé les problèmes fondamentaux et proposé une grande variété de solutions tout à fait représentatives de ce qui se fait actuellement dans le domaine.

2.3 Approches basées sur la modélisation des connaissances ou du raisonnement

2.3.1 Introduction

Le problème de l'ambiguïté du sens des mots ayant été décrit comme AI-complet, il ne peut être résolu que si une intelligence artificielle est envisageable. Un grand nombre de méthodes en Intelligence Artificielle apparaissent au début des années 1960. Elles sont appliquées au problème de l'ambiguïté du sens des mots au sein de gros systèmes dont le but est la compréhension sans limitation des langues. Ces théories, bien dans l'esprit de l'époque, sont basées sur la compréhension et la modélisation du langage humain et utilisent parfois des connaissances détaillées en syntaxe et en sémantique.

En Intelligence Artificielle, il existe deux approches fondamentalement différentes, la première est de type connexioniste [2], la seconde de type symbolique [3].

Approche connexioniste

L'approche la plus sûre permettant de réaliser une intelligence artificielle est peut-être l'approche connexioniste. C'est elle qui se rapproche le plus du fonctionnement effectif du cerveau humain. Ce type d'approche est dite émergente et s'oppose dans son concept aux approches symboliques. Elle peut se résumer de la façon suivante :

- le cerveau est le siège de la pensée et peut être totalement décrit par des mécanismes chimiques et physiques [4] ;
- il n'y a aucune impossibilité théorique quant à la reconstruction d'un cerveau humain à l'aide de composants électroniques ; chaque neurone pourrait être remplacé par un équivalent électronique remplissant les mêmes fonctions [5].

Approche symbolique

Pour les partisans de l'approche symbolique, il est bien exact que le cerveau fonctionne comme décrit par l'approche connexioniste à un niveau élémentaire. L'approche symbolique consiste à se placer à un niveau supérieur de description permettant d'avoir une vision symbolique globale du fonctionnement. La thèse de l'IA forte prétend donc qu'il existe un modèle symbolique plus simple que le modèle connexioniste, permettant d'obtenir les mêmes résultats. Une conséquence du théorème de Chaitin qui énonce qu'il est impossible de prouver que le modèle a atteint la taille minimale est que la proposition « une IA forte peut exister » est soit vraie soit indémontrable.

2. Également désignée par le terme IA faible.

3. Également désignée par le terme IA forte.

4. Cette opinion est aujourd'hui acceptée par la plupart des scientifiques, à l'exception de ceux défendant l'origine divine de l'intelligence.

5. Cette hypothèse est déjà plus difficile à accepter car elle apparaît comme parfaitement irréaliste compte tenu de la technologie actuelle et de notre connaissance encore incomplète du cerveau.

2.3.2 Approches connexionistes

Les travaux des psycholinguistes, dans les années 1960-1970, introduisent le concept d'amorçage sémantique (Meyer & Schvaneveldt, 1971) : si nous activons un concept dans la mémoire lexicale, celui-ci va activer tous les autres concepts qui lui sont sémantiquement reliés, donc il facilite le traitement de ces autres concepts.

Cette conception de la mémoire sémantique se retrouve dans les modèles de *diffusion de l'activation* (Collins & Loftus, 1975 ; Anderson, 1976, 1983) où les concepts activés dans un réseau sémantique se diffusent à des nœuds voisins. L'activation s'amenuise au fur et à mesure de sa diffusion, mais certains nœuds, recevant une activation de plusieurs sources, se renforcent progressivement. McClelland et Rumelhart (1981) complètent le modèle en introduisant la notion d'*inhibition* entre nœuds. Ainsi, l'activation d'un nœud du réseau peut inhiber, plutôt qu'activer, certains de ses voisins. Cette approche est également reprise par Feldman et Ballard (1982).

Le réseau sémantique de Quillian (cf. paragraphe *Réseaux sémantiques* de la section 2.3.3) correspond à la première implémentation de la diffusion de l'activation au sein d'un réseau dans le but de désambiguïser le sens des mots. Dans ces deux représentations, chaque nœud du réseau représente un mot ou un concept spécifique [6]. Un modèle similaire est implémenté par Cottrell et Small (1983) (voir aussi Cottrell, 1985).

Waltz et Pollack (1985) et Bookman (1987) implantent manuellement dans leur réseau des ensembles de micro-attributs sémantiques correspondant à des distinctions sémantiques fondamentales (animé/inanimé, comestible/non comestible, dangereux/non dangereux, etc.), à des durées (seconde, minute, heure, jour, etc.), à des lieux (ville, pays, etc.) et à d'autres distinctions similaires. Dans Waltz et Pollack, ces ensembles de micro-attributs sémantiques doivent être activés manuellement par l'utilisateur pour activer un contexte de désambiguïsation pour les mots à désambiguïser à venir. Bookman propose un processus dynamique qui permet d'activer automatiquement les ensembles de micro-attributs en utilisant le texte qui précède les mots à traiter.

Dans les modèles précédents, les nœuds correspondent à un mot ou à un concept spécifique. Il existe aussi des modèles distribués (Kawamoto, 1988). Dans de tels modèles, les nœuds n'ont pas de valeur symbolique particulière, et l'activation d'un concept ou d'un mot consiste en l'activation d'un ensemble, parfois appelé patron, de nœuds. Ce type de modèle est confronté à un problème de mise en œuvre important : il ne peut être construit sans passer par une phase d'apprentissage nécessitant de nombreux exemples de désambiguïsation.

2.3.3 Approches symboliques

Réseaux sémantiques

Les réseaux sémantiques sont développés dans les années 50 et très vite appliqués au problème de la représentation du sens des mots. Masterman (1961), qui travaille dans le domaine de la traduction automatique, les utilise pour modéliser les concepts fondamentaux du langage. Dans une telle approche, la désambiguïsation est intrinsèque. Elle se fait en choisissant la représentation du groupe de nœuds le plus compact dans le réseau.

6. Les méthodes symboliques, comme celle de Quillian, implémentent la propagation par l'intermédiaire d'un mécanisme de jeton. Ce mécanisme est très différent de celui utilisé dans un réseau de neurones artificiels, dont le premier développement complet est le célèbre perceptron de Rosenblatt, qui cherche à modéliser le fonctionnement réel des réseaux de neurones naturels et s'inspire des résultats obtenus en neurologie.

En s'appuyant sur les travaux de Masterman (1961) et de Richens (1958), Quillian (1961, 1962a, 1962b, 1967, 1968, 1969) construit un réseau liant les mots et les concepts dont les liens typés indiquent une simple association entre mots ou une relation sémantique particulière. Ce réseau est construit à partir des définitions d'un dictionnaire, puis est enrichi manuellement. Quand deux mots sont présentés au réseau, si plusieurs concepts sont rattachés à un même mot, la désambiguïsation se fait en choisissant le concept intervenant dans le plus court chemin reliant ces deux mots.

Hiro, Wu et Furugori (1996) utilisent un réseau sémantique basé sur des corpus. Le réseau est donc construit à l'aide de corpus du domaine dans lequel il travaille. Le choix du sens d'un mot ambigu se fait en cherchant la définition la plus compatible avec l'occurrence du mot en question du point de vue du réseau sémantique.

Structures conditionnelles

Très vite les recherches s'orientent vers l'exploitation de structures conditionnelles contenant des informations sur les mots et sur leur rôle et leurs relations avec les autres mots au sein d'une même phrase. Par exemple, Hayes (1976, 1977a, 1977b, 1978) utilise une combinaison d'un réseau sémantique et de structures conditionnelles. Le réseau est constitué de nœuds représentant des sens de mots et de liens représentant des sens de verbes. Les structures conditionnelles imposent des relations de type *est un* ou *fait partie de* dans le réseau. Dans le cas d'homonymie, le réseau de Hayes est relativement efficace, mais il ne l'est pas pour d'autres types de polysémie.

Contraintes de sélection et sémantique préférentielle

Un grand nombre d'approches utilisant les contraintes de sélection a vu le jour à partir de Katz et Fodor (1964).

La sémantique préférentielle de Wilks (1968, 1969, 1973, 1975b, 1975d, 1975a, 1975c, 1998) reprend les contraintes de sélection de Katz et Fodor à ceci près que ces contraintes ne sont plus obligatoires mais plutôt préférentielles. Elle définit des contraintes de sélection pour les adjectifs, les prépositions et les verbes. Pour les verbes, elle spécifie les classes sémantiques probables de leur sujet, voire de leurs objets. Par exemple, le sens du verbe *drink* inclut que celui-ci a pour préférences un agent *animé* et un objet *liquide*. Les sens des noms sont décrits par un ensemble de propriétés basiques (*humain*, *animé*, *concret*, *abstrait*, etc.) qui déterminent leur place dans un réseau sémantique. Les termes des différentes catégories syntaxiques sont représentés dans des réseaux sémantiques séparés. C'est la première approche spécialement étudiée pour le problème de la désambiguïsation lexicale. L'analyse sémantique d'une phrase consiste à apparier les éléments en présence avec des contraintes imposées par le verbe. Pour le traitement d'un énoncé, le système recherche les structures sémantiques possibles et retient la structure qui satisfait le plus grand nombre de préférences. Quand les seules interprétations possibles violent toutes des classes ou des préférences de classe, l'interprétation retenue est celle qui pose le moins de conflits. La sémantique préférentielle permet ainsi de prendre en compte certaines métaphores.

Boguraev (1979) montre que la sémantique préférentielle ne permet pas de résoudre la polysémie verbale. Pour remédier à cet inconvénient, il utilise le formalisme de Wilks pour la représentation sémantique mais y ajoute une représentation syntaxique. C'est en tenant compte de ces deux informations qu'il cherche à lever l'ambiguïté sur le sens des mots. Comme tous les systèmes précédents, cette nouvelle approche se limite à la phrase isolée. Aucune information sur le domaine ou le thème n'est donc prise en compte. Aussi, certaines formes de désambiguïsation sont difficiles, voire impossibles, à réaliser.

Experts en mots

Les *parsers* experts en mots (Small, 1980, 1983 ; Small & Rieger, 1982 ; Adriaens, 1986a, 1986b, 1989 ; Adriaens & Small, 1988) constituent une approche de la compréhension de la langue, capable de désambiguïsation, très différente des autres approches. En effet, elle postule que la compréhension humaine du langage est basée sur une connaissance approfondie des mots plutôt que sur une connaissance approfondie des règles. Le système modélise ce que la théorie sur la compréhension du langage par les hommes suppose : une coordination de l'échange d'information sur la syntaxe et la sémantique entre différents experts en mots. En fait, chaque expert, rattaché à un mot, constitue une entité distincte. Elle reçoit les informations fournies par le contexte et par d'autres experts en mots pour aboutir à un sens unique qui est, ensuite, ajouté à la représentation sémantique de la phrase. Le problème est que chaque expert en mot doit être extrêmement complexe pour accomplir sa tâche. Mais il faut reconnaître que cette tâche va bien au-delà de la désambiguïsation.

Mots *polaroïd*

Hirst (1987) présente une architecture distribuée requérant moins d'opérations manuelles qu'un *expert en mot*. Plutôt que de créer un module d'*expert en mot* distinct pour chaque mot, Hirst crée des modules de mots *polaroïd* pour chaque catégorie syntaxique. Les informations relatives à un mot particulier sont contenues dans une base de connaissances. Pour traiter un mot, le mot *polaroïd* approprié est instancié avec les informations, relatives au mot traité, contenues dans la base de connaissances. Cette manière de séparer l'information spécifique à chaque mot de l'information commune à tous les mots d'une même catégorie syntaxique, permet une économie importante en terme de complexité de la représentation ainsi qu'en terme de codage manuel. Hirst utilise également des réseaux de structures avec une méthode de désambiguïsation analogue à celle de Quillian (1968, 1969). Il ajoute en plus un moyen d'éliminer progressivement les sens inadéquats en combinant les informations syntaxiques issues du *parser* et les relations sémantiques contenues dans le réseau. Parfois un seul sens subsiste, mais il arrive que, dans le cas de métaphores ou de métonymies, aucun sens ne subsiste.

Raisonnements de sens commun

Le système de compréhension du langage élaboré par Dahlgren (1988) possède un module de désambiguïsation qui utilise plusieurs types d'informations : phrases isolées, informations syntaxiques (primaires et contraintes de sélection) et raisonnements de sens commun. Ce dernier module, étant très gourmand en temps de calcul, n'est invoqué que si les deux premiers ont échoué. Dahlgren observe que la moitié des désambiguïsations peut s'effectuer en utilisant les phrases isolées et les informations syntaxiques tandis que l'autre moitié nécessite l'utilisation des raisonnements de sens commun. Le raisonnement conduit parfois à rechercher des ancêtres communs aux mots du contexte à travers une ontologie. Ces travaux anticipent les résultats de Resnik (1993a, 1995) en montrant que les similarités ontologiques constituent un moyen de désambiguïsation puissant. Dahlgren observe également que les contraintes de sélection sur les verbes constituent une importante source de désambiguïsation pour les noms.

Approche statistique des contraintes de sélection

Resnik (1993a, 1997) introduit une approche statistique des contraintes de sélection. Son approche possède l'avantage d'être générale dans le sens où elle n'est pas restreinte à une petite partie du langage. De plus, cette approche ne nécessite pas de corpus étiqueté pour son apprentissage, elle s'affranchit donc de ce goulet d'étranglement.

2.3.4 Limitations

Toutes les approches qui utilisent des bases de connaissances spécifiques construites manuellement, comme par exemple la sémantique préférentielle de Wilks, les structures conditionnelles de Hayes et Hirst, les experts en mots de Small, Rieger et Adriaens ou les ensembles de micro-attributs de Bookman, Waltz et Pollack, ne sont pas très prometteuses pour tout système dont l'ambition est une couverture importante de la langue (Wilks & Stevenson, 1997a). Le principal obstacle à la généralisation de ces méthodes est la difficulté et le coût du codage manuel de l'immense connaissance nécessaire à la désambiguïsation de textes non restreints. Ces méthodes se cantonnent donc à de petits sous-ensembles du langage et souvent se limitent à de petits contextes (une seule phrase par exemple). Il est difficile d'évaluer leurs performances en situation réelle, c'est-à-dire sur des textes de taille importante couvrant une grande partie du langage.

2.4 Approches utilisant des bases de connaissances informatisées (approches exogènes)

2.4.1 Introduction

Toutes les méthodes utilisées dans les années 1970 et 1980 sont intéressantes d'un point de vue théorique, mais assez limitées dans la pratique. La recherche en désambiguïsation lexicale connaît un tournant dans les années 1980 avec l'apparition de ressources informatisées à grande échelle comme les dictionnaires, les thésaurus, les grands corpus, ainsi qu'avec l'augmentation importante de la puissance de calcul des ordinateurs modernes. Des efforts se portent alors sur l'extraction automatique des connaissances de telles ressources. Les méthodes basées sur des théories linguistiques et destinées à un traitement global des phénomènes linguistiques sont petit à petit abandonnées au profit de théories basées sur des méthodes empiriques s'intéressant à des tâches intermédiaires comme la désambiguïsation lexicale.

2.4.2 Dictionnaires informatisés

Après les thèses de Amsler (1980) et de Michiels (1982), les *dictionnaires informatisés* (*Machine-readable dictionaries* en anglais) deviennent une source d'information très appréciée pour le traitement automatique des langues. Dans les années 80, un des principaux domaines d'activité consiste à extraire automatiquement des bases de connaissances lexicales et sémantiques de grande taille à partir de dictionnaires informatisés (Michiels, Mullenders & Noël, 1980 ; Calzolari, 1984 ; Chodorow, Byrd & Heidorn, 1985 ; Markowitz, Ahlswede & Evens, 1986 ; Byrd *et al.*, 1987 ; Nakamura & Nagao, 1988 ; Klavans, Chodorow & Wacholder, 1990 ; Wilks *et al.*, 1990 ; etc.). Ces travaux contribuent grandement aux recherches dans le domaine de la sémantique lexicale, mais il apparaît que l'objectif initial, c'est-à-dire l'extraction automatique de bases de connaissances de grande taille, n'est pas atteint. Actuellement, l'unique base

de connaissances lexicales de taille importante et disponible pour tous est WordNet. Elle a été constituée manuellement. D'autre part, Ide et Véronis (1991, 1993a, 1993b) montrent qu'il est très difficile d'extraire de manière automatique des relations aussi simples que l'hyperonymie. Cela est en grande partie dû à l'inconsistance même des dictionnaires [7] mais aussi au fait que ces dictionnaires sont avant tout conçus pour les besoins des hommes et non pour ceux des machines.

Malgré toutes ces imperfections, les dictionnaires informatisés constituent une source d'information existante en ce qui concerne le sens des mots, aussi deviennent-ils rapidement un élément de base de la recherche en désambiguïsation lexicale. Les méthodes utilisées cherchent à s'affranchir des problèmes précités en combinant l'utilisation directe du texte de chaque définition avec des techniques suffisamment robustes pour minimiser les inconsistances des dictionnaires. Toutes ces méthodes reposent sur la constatation suivante : quand plusieurs mots sont cooccurrents, le sens le plus probable pour chacun de ces mots est celui qui maximise ses relations avec le sens des mots cooccurrents.

Lesk (1986) crée une base de connaissances qui comporte pour chaque sens d'un mot la liste des mots apparaissant dans sa définition. La désambiguïsation est réalisée en choisissant le sens du mot pour lequel la liste associée comporte le plus de similitudes avec la liste des cooccurrences de ce mot dans le texte. Cette méthode permet de trouver le bon sens dans 50% à 70% des cas, en utilisant une palette de sens assez fine comme celle que procure un dictionnaire classique pour étudiant. Cette méthode est très sensible aux mots qui se trouvent dans chaque définition : la présence ou l'absence d'un mot donné peut radicalement changer le résultat. La méthode de Lesk sert tout de même de base pour la plupart des travaux subséquents, en désambiguïsation, utilisant des dictionnaires informatisés.

Wilks *et al.* (1990) cherchent à améliorer les connaissances associées à chaque sens en calculant les fréquences des cooccurrences dans les définitions. Ils en dérivent plusieurs mesures du degré de corrélation entre mots. Cette mesure est utilisée, dans le cadre d'une approche vectorielle, pour mettre en relation chaque mot et son contexte. Les expériences sur le mot *bank* montrent une performance de 45% sur l'identification du sens correct et une performance de 90% sur l'identification de l'homographe correct.

La méthode de Lesk est étendue par Véronis et Ide (1990) en générant un réseau de neurones construit à partir des définitions du dictionnaire anglais Collins (CED). Dans ce réseau, chaque mot est connecté à ses sens, eux-même connectés aux mots apparaissant dans leur définition qui sont à leur tour connectés à leurs sens, etc. Une expérience menée sur 23 mots polysémiques montre un taux de réussite de 71,7% en utilisant les distinctions de sens relativement fines du CED.

De nombreux auteurs (Krovetz & Croft, 1989 ; Guthrie, Guthrie, Wilks & Aidinejad, 1991 ; Slator, 1992 ; Cowie, Guthrie & Guthrie, 1992 ; Janssen, 1992 ; Braden-Harder, 1993 ; Liddy & Paik, 1993 ; etc.) tentent d'améliorer les performances en utilisant les informations supplémentaires de la version électronique du dictionnaire de l'anglais contemporain Longman (LDOCE), en particulier les codes sémantiques [8] et les catégories de sujet [9] disponibles pour chaque sens. Les codes [10] de LDOCE véhiculent, en plus des codes sémantiques, des restrictions de type au niveau des noms et des adjectifs et au

7. Cette inconsistance est bien connue des lexicographes (Atkins & Levin, 1991 ; Kilgarriff, 1994).

8. LDOCE comporte 10 codes sémantiques (*semantic code* en anglais) par exemple, le code *object* possède les valeurs *adstract*, *plant*, *liquide*, *human* etc.

9. LDOCE comporte 124 catégories de sujet (*subject categories* en anglais) majeures, par exemple *economics*, *engineering*, etc. Certaines catégories contiennent des sous-catégories. LDOCE comporte au total 369 sous-catégories.

10. *Box codes* dans le texte.

niveau des arguments des verbes. Ainsi, Guthrie *et al.* (1991) reprennent la méthode de Lesk en imposant la correspondance au niveau des catégories de sujet. L'utilisation des codes de LDOCE est toutefois problématique car ces codes ne sont pas systématiques (Fontenelle, 1990). De plus, Braden-Harder (1993) montre que la simple correspondance des codes et des catégories de sujet n'est pas suffisante pour lever toute ambiguïté.

L'inconsistance dans les dictionnaires ne constitue pas la seule, voire même la plus importante limitation en désambiguïsation. En effet, si les dictionnaires contiennent des informations détaillées au niveau lexical, ils manquent cruellement d'informations pragmatiques, alors que celles-ci constituent une bonne source pour la désambiguïsation (Hobbs, 1987). Par exemple, les liens entre *cendre* et *tabac*, *cigarette* ou *cendrier*, dans un réseau tel que celui de Quillian, sont très indirectes alors que ces trois mots sont fréquemment en cooccurrence avec le mot *cendre* dans les corpus. Aussi n'est-il pas étonnant que les corpus deviennent, par la suite, la première source d'information en désambiguïsation (cf. section 2.5).

2.4.3 Thésaurus

Les *thésaurus* (*thesauri* en anglais) fournissent des informations sur les relations entre les mots et plus particulièrement les synonymes. Le thésaurus international ROGET (ROGET's *International Thesaurus*) est informatisé dans les années 1950. Il est utilisé dans de nombreuses applications comme la traduction automatique (Masterman, 1957), la recherche d'informations (Sparck Jones, 1964, 1986 ; Voorhees, 1993) et l'analyse de contenu (Sedelow & Sedelow, 1969, 1986, 1992). Le thésaurus ROGET est hiérarchisé en huit niveaux de concepts de granularité croissante. En fait, les occurrences d'un même mot au sein des différentes catégories correspondent approximativement aux différents sens de ce mot (Yarowsky, 1992). Des mots différents au sein d'une même catégorie sont proches sémantiquement.

Masterman, en 1957, est le premier à utiliser le thésaurus ROGET dans le domaine de la désambiguïsation. Bryan (1973, 1974) [11] montre qu'il est possible de distinguer les homographes en utilisant une métrique sur les liens définis par des chaînes dérivées du thésaurus ROGET. Patrick (1985) utilise ensuite ce thésaurus pour désambiguïser le sens des verbes de manière efficace.

Yarowsky (1992) génère des classes de mots dérivées des catégories de concept de premier niveau (*i.e.* de granularité grossière) du thésaurus ROGET. Chaque classe de mots est constituée des cooccurrences les plus fréquentes, observées dans un corpus [12], des mots contenus dans la catégorie de concept associée. Certains mots appartiennent à plusieurs catégories de concept. Les cooccurrences des mots contenus dans une catégorie de concept donnée sont donc bruitées. Cependant, ce bruit reste faible au regard du signal qui se dégage de la catégorie commune à tous les mots. Ce bruit est filtré par un seuil de fréquence minimum des cooccurrences considérées pour constituer la classe de mots associée à une catégorie de concept. Un mot polysémique est désambiguïsé en recherchant, par la règle de BAYES, la classe la plus probable compte tenu des mots du contexte du mot à désambiguïser. Dans cette expérience, la précision atteinte est de 92% avec une moyenne de trois sens par mot. Yarowsky observe que cette méthode permet essentiellement d'extraire les informations sur le thème et qu'elle est donc essentiellement efficace pour les noms.

Le problème des thésaurus est semblable à celui des dictionnaires (cf. section 4.2.1) c'est-à-dire qu'ils manquent de cohérence et sont, avant tout, destinés à être utilisés

11. Voir aussi Sedelow et Donna (1988) pour des travaux similaires.
12. Le corpus utilisé est le corpus électronique *Grolier's Encyclopedia* de 1991.

par des humains. Toutefois, ils mettent à disposition un riche réseau d'associations de mots et un ensemble de catégories sémantiques potentiellement exploitables pour le traitement du langage, mais ils n'ont pas encore été utilisés de manière intensive dans le domaine de la désambiguïsation du sens des mots.

2.4.4 Lexiques informatiques

Au milieu des années 1980, de nombreux efforts se portent sur la réalisation manuelle de grandes bases de connaissances informatiques appelées *lexiques informatiques* (*Computational lexicons* en anglais), par exemple WordNet (Miller, Beckwith, Fellbaum, Gross & Miller, 1990 ; Fellbaum, 1998), cyc (Lenat & Guha, 1989), acquilex (Briscoe, 1991), comlex (Grishman, MacLeod & Meyers, 1994, 1999), etc.

Il existe deux approches distinctes à la construction de ces lexiques sémantiques. La première, dite énumérative, fournit explicitement tous les sens d'un mot. La seconde, dite générative, fournit des informations sémantiques générales pour chaque mot ; il faut alors utiliser des règles génératives pour dériver des informations précises sur les sens (Fellbaum, 1999).

Lexiques énumératifs

Entre tous les *lexiques énumératifs* (*Enumerative lexicons* en anglais), WordNet est le plus connu et le plus utilisé pour la désambiguïsation automatique en anglais. Des versions de WordNet pour plusieurs langues européennes sont en cours de développement (Vossen, 1998 ; Sutcliffe *et al.*, 1996a, 1996b).

L'avantage de WordNet réside dans la diversité des informations qu'il contient. En effet, il comporte les ressources qui se trouvent dans les dictionnaires (définition de chacun des sens) et les thésaurus (ensembles de synonymes représentant les concepts lexicaux organisés en une hiérarchie conceptuelle), mais également des liens entre les mots selon diverses relations sémantiques telles que l'hyponymie, l'hyperonymie, l'antonymie et la méronymie, etc. Une autre raison de la grande popularité de WordNet est qu'il s'agit d'une ressource lexicale de grande couverture librement et gratuitement utilisable. Aussi, malgré les défauts de ce lexique, la division sémantique et les relations lexicales de WordNet ont certainement un impact important sur les recherches actuelles et à venir (Kilgarriff, 1997a).

Les premières tentatives d'exploitation de WordNet pour la désambiguïsation sont réalisées dans le domaine de la recherche d'informations. En utilisant les liens d'hyponymie entre les noms de WordNet, Voorhees (1993) conçoit une structure appelée *hood* dans le but de représenter les catégories sémantiques, dans le même esprit que les catégories du thésaurus Roget. Un *hood*, pour un mot donné, est défini comme étant le plus grand sous-graphe connecté contenant ce mot. Les résultats que Voorhees obtient indiquent que sa technique n'est pas utilisable pour distinguer les sens fins de WordNet. Dans une étude similaire, Richardson et Smeaton (1994) créent une base de connaissances à partir de la hiérarchie de WordNet. Ils appliquent une fonction de similarité, développée par Resnik (1995), pour lever l'ambiguïté dans le cadre de la recherche de documents. Bien qu'ils ne fournissent pas d'évaluation de leur méthode, ils prétendent que les résultats sont prometteurs.

Sussna (1993) calcule une distance sémantique entre les noms rencontrés dans le texte. Cette distance prend en compte différentes relations (synonymie, hyperonymie, etc.) de WordNet. L'hypothèse est que, pour un ensemble de noms dont les occurrences sont proches dans le texte, le sens de ces noms est celui qui minimise la distance

entre eux. Les résultats de cette méthode sont significativement meilleurs que le hasard. Cette méthode est également intéressante dans le sens où il en existe peu qui utilisent d'autres relations que la relation *IS-A* de WordNet.

Resnik (1993a, 1993b, 1995) explore une mesure de similarité sémantique construite à partir de la taxinomie *IS-A* des noms de WordNet. Le principe qui sous-tend cette mesure est que plus deux mots sont sémantiquement proches, plus le concept qui les subsume est spécifique. Resnik justifie sa mesure par rapport à un calcul classique de longueur de chemin (Rada, Mili, Bicknell & Blettner, 1989 ; Sussna, 1993 ; Lee, Kim & Lee, 1993) en remarquant que les liens de la taxinomie *IS-A* de WordNet ne correspondent pas à une mesure de distance uniforme. La méthode de Resnik approche les performances humaines pour la désambiguïsation en utilisant le niveau de sens fin de WordNet. Comme dans la plupart des travaux ici présentés, seuls les noms sont traités.

Lexiques Génératifs

Pustejovsky (1991) critique la solution trop simpliste consistant à faire un choix parmi un inventaire de sens préexistants ne permettant pas de rendre compte des extensions systématiques des emplois. L'alternative qu'il offre, basée sur les *lexiques génératifs* (*Generative lexicons* en anglais), réside dans une construction dynamique du sens qui n'est pas sélectionné parmi un ensemble de possibles, mais engendré par des règles capturant des régularités dans la création du sens (comme la métonymie par exemple). Buitelaar (1997) décrit l'utilisation de CORELEX, pour un étiquetage sémantique sous-spécifié (voir également Pustejovsky, Boguraev & Johnston, 1995).

Viegas, Mahesh et Nirenburg (1998) décrivent une approche similaire entreprise dans le cadre de la traduction automatique (voir également Mahesh, Nirenburg & Beale, 1997 ; Mahesh *et al.*, 1997). Ils utilisent des lexiques syntaxiques et sémantiques qui fournissent une information détaillée au niveau des restrictions de sélection pour les mots de la phrase et recherchent une ontologie fortement connectée pour le sens du mot cible qui satisfasse le mieux ces contraintes. Ils obtiennent une précision de 97%. Comme dans CORELEX, le lexique comme l'ontologie sont construits manuellement et, par là même, difficilement généralisables pour une couverture plus grande de la langue. Cependant, Buitelaar propose des méthodes permettant de générer automatiquement les entrées de CORELEX à partir d'un corpus pour construire un lexique sémantique spécifique à un domaine. Il démontre ainsi la possibilité d'adresser des ressources de ce type à grande échelle.

2.5 Approches basées sur corpus (approches endogènes)

2.5.1 Introduction

Les corpus sont utilisés très tôt pour des travaux concernant le sens des mots (Palmer, 1933 ; Eaton, 1940 ; Zipf, 1945 ; Thorndike, 1948 ; Lorge, 1949). Les corpus étant de grands ensembles d'exemples, leur utilisation va de pair avec les méthodes empiriques et statistiques.

Mais par la suite, les traitements statistiques sur corpus sont dénigrés au profit des règles linguistiques formelles. Pourtant, pendant cette période, plusieurs corpus de taille importante sont constitués comme le corpus BROWN (Kucera & Francis, 1967), le *Trésor de la Langue Française* (TLF, Imbs, 1971), le corpus *Lancaster-Oslo-Bergen* (Johansson,

1980), etc. Dans le domaine du traitement automatique des langues, en 1966, le rapport ALPAC (1966) préconise une utilisation intensive des corpus pour créer des grammaires et des lexiques à grande couverture. L'utilisation des corpus continue toutefois à se faire rare. Dans ce contexte, l'utilisation des corpus et des méthodes empiriques faite par Weiss (1973) et par Kelly et Stone (1975), dans le domaine de l'extraction automatique de connaissances pour la désambiguïsation lexicale, semble innovante.

Le principe de base de l'acquisition de connaissances à partir des corpus et pour la désambiguïsation lexicale est très simple. En étudiant un grand nombre de contextes des occurrences de chacune des lexies d'un mot polysémique, il doit être possible d'identifier statistiquement des indices récurrents se démarquant (*i.e.* des indices *saillants*) pour chacune des lexies.

Kelly et Stone (1975) travaillent sur un corpus d'un demi-million de mots. Avec l'aide de leurs étudiants, ils construisent manuellement un modèle de désambiguïsation pour 1815 mots du corpus possédant une fréquence de plus de 20 occurrences. Les modèles sont construits en observant un contexte de quatre mots à gauche et à droite pour chacune des occurrences des 1815 mots dans le corpus. Chacun de ces modèles contient un ensemble de règles basées sur la morphologie du mot, sur la présence de mots saillants dans le contexte et sur la classe sémantique (déterminée manuellement) des mots du contexte.

2.5.2 Approches basées sur corpus étiquetés

Dans l'hypothèse idéale où un corpus lexicalement étiqueté est disponible, l'approche la plus adéquate est probablement de mettre en œuvre des techniques d'apprentissage supervisé. Hélas, cette approche se trouve confrontée au problème crucial du manque de corpus lexicalement étiquetés (Gale *et al.*, 1992b). L'étiquetage manuel de ces corpus demande énormément de temps et donc d'argent. Certains corpus commencent tout juste à exister, cependant ils sont relativement récents (les études passées n'ont pas pu en profiter), ils sont parfois de petite taille et ne couvrent pas tous les domaines ni toutes les langues. Cet handicap a longtemps compromis, et compromet souvent encore, la pertinence des approches basées sur corpus étiquetés.

Weiss (1973) est l'un des premiers à s'intéresser à l'apprentissage automatique de règles de désambiguïsation à partir de corpus étiquetés. Il utilise deux types de règles : des règles recherchant la présence de mots saillants dans un contexte de taille donnée et des règles recherchant la présence d'un mot saillant à une position donnée par rapport au mot à désambiguïser.

L'utilisation des corpus revient à la mode dans les années 1980. Black (1988) mène une étude portant sur cinq mots. Il effectue un étiquetage manuel de 2000 occurrences de chacun de ces mots. 1500 occurrences sont utilisées pour réaliser des apprentissages et 500 pour des évaluations des méthodes de désambiguïsation. Black compare dans son étude trois méthodes de désambiguïsation. La première est très proche de celle de Weiss. La seconde inclut en plus des catégories thématiques manuellement spécifiées. La troisième est basée sur les catégories de sujet (par exemple *economics*, *engineering*) du dictionnaire LDOCE.

À partir de là, les méthodes d'apprentissage automatique supervisé sont largement utilisées (Bruce & Wiebe, 1994b, 1994a ; Pedersen, Bruce & Wiebe, 1997 ; Pedersen & Bruce, 1997b ; Mooney, 1996 ; Ng & Lee, 1996 ; Ng, 1997 ; El-Bèze, Loupy & Marteau, 1998 ; El-Bèze, Michelon & Pernaud, 1999 ; Escudero, Marquez & Rigau, 2000b, 2000c ; Yarowsky, 2000 ; Yarowsky, Cucerzan, Florian, Schafer & Wicentowski, 2001 ; Agirre

& Martinez, 2000 ; Pedersen, 2001, 2002 ; Ng & Lee, 2002 ; etc.) [13]. Dans une approche supervisée, un algorithme extrait automatiquement les connaissances nécessaires pour la désambiguïsation à partir d'un grand corpus lexicalement étiqueté dans lequel les mots ont été étiquetés manuellement selon les lexies d'un dictionnaire donné (LDOCE, WORDNET, etc.). Cette phase d'extraction automatique des connaissances est appelée apprentissage. À l'issue de cette phase, l'algorithme de désambiguïsation est capable d'assigner la lexie adéquate aux mots apparaissant dans une nouvelle phrase en se basant sur les connaissances acquises durant la phase d'apprentissage. Au cœur des approches supervisées basées sur corpus étiquetés se trouvent donc des techniques de classification supervisée (cf. chapitre 6). Ces techniques ont besoin de corpus manuellement étiquetés au niveau lexical pour la phase d'apprentissage mais également pour l'évaluation de leurs performances.

Les corpus étiquetés constituent un réservoir abondant de connaissances sur la langue. Concernant la langue anglaise, nous pouvons citer le corpus SEMCOR (Fellbaum, 1998) contenant 250 000 mots étiquetés selon l'inventaire des lexies de WORDNET. Ce corpus est constitué de textes extraits du corpus BROWN et de la nouvelle « *The Red Badge of Courage* ». Nous pouvons également citer le corpus DSO (Ng & Lee, 1996) qui contient 192 800 occurrences des 121 noms et 70 verbes les plus fréquents et les plus ambigus de l'anglais. Ces occurrences sont étiquetées selon l'inventaire des lexies de WORDNET. Ce corpus contient également des extraits du corpus BROWN. Il ne faut pas oublier non plus les corpus de référence rendus disponibles dans le cadre des campagnes d'évaluation SENSEVAL-1 (HECTOR, Kilgarrif & Rosenzweig, 2000), SENSEVAL-2 et bientôt SENSEVAL-3.

Cependant, l'extraction des connaissances utiles à la désambiguïsation lexicale à partir de corpus étiquetés constitue une tâche difficile posant deux problèmes majeurs. Le premier problème est, comme nous l'avons déjà mentionné, le manque de corpus lexicalement étiquetés. En raison de ce manque, la plupart des études sont menées sur un nombre limité de mots, voire sur un seul. Trois méthodes permettant de contourner ce problème sont présentées dans les sections 2.5.3, 2.5.4 et 2.5.5. Une autre solution est tout simplement la constitution de cette ressource manquante. Ce travail est en bonne voie pour une langue comme l'anglais, mais actuellement inexistant pour la plupart des autres langues et pour le français en particulier (Kilgarriff, 1998a). Le second problème ne concerne pas que les corpus étiquetés et est probablement le plus préoccupant. Il réside dans la dispersion des données et fait l'objet de la section 2.5.6.

2.5.3 *Pseudo-mots* contre la carence en corpus étiquetés

Des études (Brown, Pietra, Pietra & Mercer, 1991 ; Dagan, Itai & Schwall, 1991 ; Dagan & Itai, 1994 ; Gale, Church & Yarowsky, 1992d, 1992b ; Yarowsky, 1993 ; etc.) contournent le problème du manque de corpus lexicalement étiquetés en mettant en œuvre toute une variété de subterfuges permettant d'utiliser des techniques classiques de classification supervisée tout en s'affranchissant de la phase d'étiquetage manuel de corpus préalable. Ces subterfuges s'appuient sur des *pseudo-mots*, générés automatiquement, qui possèdent une forme d'ambiguïté et qui véhiculent l'information sur cette ambiguïté pour l'apprentissage et l'évaluation des algorithmes de classification supervisée. Yarowsky (1993) donne une liste de subterfuges permettant de créer de tels *pseudo-mots* :

1. fusion de deux mots quelconques possédant la même étiquette morphosyntaxique

13. Ces méthodes d'apprentissage automatique supervisé sont également utilisées sur des *pseudo-mots* qui font l'objet de la section suivante (2.5.3).

en un seul en gardant l'information du mot d'origine (par exemple en remplaçant dans un corpus les occurrences des mots *guerilla* et *reptil* par le *pseudo-mot guerilla/reptil*) ;
2. utilisation de mots possédant des traductions différentes dans une autre langue ;
3. fusion de deux mots homophones comme *censor/sensor* ou *cue/queue* ;
4. fusion de deux mots régulièrement confondus par les logiciels d'OCR comme les paires minimales *terse/tense* ou *cookie/rookie*.

Yarowsky (1993) justifie le subterfuge 1 en remarquant que deux mots possédant la même étiquette morphosyntaxique peuvent très bien être représentés par un mot unique dans une autre langue. Le problème consistant à discriminer ces deux mots en fonction de leur contexte est bien un problème de désambiguïsation lexicale. Le subterfuge 2 est un raffinement du précédent dans lequel la fusion dans une autre langue est avérée. Cette technique, utilisée par Brown *et al.* (1991), Gale *et al.* (1992d, 1992b) et Yarowsky (1993), par exemple, nécessite de disposer de corpus parallèles de grande taille et de thèmes variés. Or, ces corpus sont rares. Pour pallier cette difficulté, Dagan *et al.* (1991) et Dagan et Itai (1994) proposent une méthode similaire qui, au lieu d'utiliser des corpus parallèles, utilise deux corpus monolingues et un dictionnaire bilingue.

Selon Yarowsky, les *pseudo-mots* sont très utiles pour développer et tester des méthodes de désambiguïsation lexicale automatique supervisée. Cependant, ces subterfuges souffrent de nombreux inconvénients. Les *pseudo-mots* de l'item 1, bien qu'utiles pour développer et tester des méthodes de désambiguïsation, sont totalement inapplicables à une tâche réelle de traitement automatique des langues (TAL) étant donné que les mots utilisés sont fabriqués et n'ont aucune existence réelle. Quant aux *pseudo-mots* de l'item 2, ils conservent généralement une partie, voire la totalité de l'ambiguïté. Par exemple, tous les sens majeurs du mot *interest* en anglais sont traduits par le mot *intérêt* en français. Ainsi, si cette approche est utile dans le domaine de la traduction automatique, elle l'est moins pour d'autres tâches du TAL. Cette remarque est également applicable aux *pseudo-mots* des items 3 et 4. Une autre limitation provient des corpus bilingues utilisés. Tout d'abord, les corpus parallèles de grande taille, comme le corpus HANSARD utilisé par Yarowsky, sont très spécialisés ce qui biaise les résultats. D'autre part, l'utilisation de corpus bilingues parallèles ne permet pas de s'affranchir de la phase de prétraitement manuelle. En effet, les traductions, comme la désambiguïsation nécessaire à la traduction, sont effectuées manuellement et nécessitent encore plus d'effort qu'un étiquetage lexical de corpus. Wilks et Stevenson (1997a) ajoutent que les recherches basées sur des méthodes de désambiguïsation lexicale automatique supervisée sont compromises par le fait que de nombreux chercheurs se focalisent sur des ambiguïtés lexicales qui ne correspondent pas à des distinctions de sens de vrais mots.

2.5.4 Approches basées sur corpus non étiquetés

Pour pallier le problème de la rareté des corpus lexicalement étiquetés, des recherches sont menées dans le but de s'affranchir de cet étiquetage (Pereira, Tishby & Lee, 1993 ; Schütze, 1992, 1998). Dans ce type d'approche, la notion de sens est généralement directement induite du corpus. Ainsi, Schütze (1992, 1998) propose une méthode basée sur le modèle des espaces de vecteurs utilisé en recherche d'informations (Salton, Wong & Yang, 1975). Dans cette approche, chaque mot est représenté par un vecteur dans un espace de grande dimension. Ces vecteurs sont automatiquement regroupés en paquets en fonction de leur degré de similitude. Chaque paquet est supposé être représentatif d'un sens.

Ces méthodes possèdent certains avantages mais véhiculent également un inconvénient majeur : les sens ne correspondent à aucun ensemble de sens bien défini. Les distinctions de sens peuvent parfois s'avérer déroutantes et sont, de plus, souvent difficilement utilisables par d'autres applications (Wilks & Stevenson, 1997a). Pour pallier ce problème, Pedersen et Bruce (1997a) proposent une technique permettant de faire correspondre ces groupes de sens aux sens d'un lexique donné. Cependant, les résultats obtenus ne sont pas encourageants et moins bons qu'en affectant le sens le plus fréquent à chaque occurrence.

2.5.5 Étiquetage incrémental de corpus

Certains chercheurs (Zernik, 1990 ; Hearst, 1991 ; Yarowsky, 1995) tentent de réduire la difficulté de la constitution de corpus étiquetés en utilisant des méthodes d'étiquetage incrémental (technique appelée *bootstrapping* en anglais).

La méthode de Zernik (1990) commence par utiliser des critères fiables facilement et manuellement identifiés pour partiellement désambiguïser le corpus. Ces critères incluent la présence de mots saillants dans un contexte et la morphologie du mot à désambiguïser. Après quoi, un apprentissage est réalisé sur les mots désambiguïsés pour effectuer une désambiguïsation des mots restants.

La méthode de Hearst (1991) consiste à étiqueter manuellement un petit ensemble d'exemples servant de données d'apprentissage à une méthode de désambiguïsation utilisée pour désambiguïser les exemples restants. Seuls les exemples dont le sens est automatiquement identifié avec une fiabilité suffisante sont effectivement désambiguïsés. Ces nouveaux exemples sont alors traités comme des exemples manuellement étiquetés pouvant à leur tour servir de données d'apprentissage.

Conformément à la méthode de Zernik, la méthode de Yarowsky (1995) commence par utiliser des critères fiables facilement et manuellement identifiés pour partiellement désambiguïser le corpus. L'étiquetage incrémental du corpus est en revanche plus sophistiqué. Comme avec la méthode de Hearst, seuls les exemples dont le sens est automatiquement identifié avec une fiabilité suffisante sont effectivement désambiguïsés. Le cycle apprentissage/étiquetage automatique est réitéré jusqu'à ce que le nombre d'exemples non étiquetés devienne faible. L'hypothèse « un sens par discours » (Gale, Church & Yarowsky, 1992c) est exploitée afin de corriger d'éventuelles erreurs de classification.

Ces méthodes d'étiquetage incrémental de corpus ne nécessitent pas l'étiquetage manuel de tout le corpus. Elles nécessitent tout de même une intervention humaine pour réaliser un étiquetage partiel dans le cas de la méthode de Hearst et pour fournir des critères fiables pour partiellement désambiguïser le corpus dans le cas des méthodes de Zernik et Yarowsky. Les critères fiables nécessaires pour partiellement désambiguïser le corpus ne sont pas toujours identifiables pour tous les mots polysémiques ou pour chacune de leurs lexies. Ces méthodes peuvent donner de bons résultats sur certains mots et pas sur d'autres. Par exemple, sur les cinq mots étudiés par Zernik, de bons résultats ne sont obtenus que sur un seul (le mot *interest*). De plus, ces méthodes semblent nécessiter de très gros corpus. Zernik utilise un corpus de 10 millions de mots et Yarowsky un corpus de 460 millions de mots.

2.5.6 Problème de la dispersion des données

L'un des problèmes majeurs auquel se confrontent les approches basées sur corpus est la densité extrêmement faible de certains phénomènes observés (*data sparseness*

en anglais). Le problème posé par cette dispersion des données est particulièrement préoccupant. En effet, en raison de la très grande diversité du vocabulaire des langues naturelles, même dans de gros corpus, la plupart des mots ne comportent que très peu d'occurrences. Les approches basées sur corpus étiquetés fondent leur désambiguïsation sur des connaissances tirées du contexte des mots à désambiguïser. Cette démarche décuple le problème de dispersion car la cooccurrence de deux mots est encore plus faible que celle d'un seul mot. En effet, quel va être le nombre d'occurrences d'un mot de densité faible dans le contexte d'un mot lui-même de densité faible ? Pire encore, quel va être le nombre d'occurrences d'un mot de densité faible dans le contexte d'une occurrence dont la lexie est elle même de densité faible parmi l'ensemble des lexies possibles d'un vocable lui-même de densité faible ? Par exemple, dans le corpus du projet SYNTSEM de six millions de mots, un mot aussi courant que *barrage* ne possède que 92 occurrences. Parmi ces 92 occurrences, la lexie 1 (« action de barrer, gêner ou empêcher physiquement le passage de quelqu'un, de quelque chose. ») n'est rencontrée qu'une fois dans le corpus. Ainsi, quelle que soit la taille du corpus utilisé, le problème de la dispersion des données reste un obstacle majeur.

Les modèles basés sur des classes tentent de pallier ce problème de dispersion en effectuant des généralisations (*i.e.* regroupements). Brown, Della Pietra, deSouza, Lai et Mercer (1992), Pereira et Tischby (1992) et Pereira *et al.* (1993) proposent des méthodes qui dérivent des classes (*i.e.* effectuent des regroupements) à partir de propriétés distributionnelles extraites du corpus. D'autres auteurs utilisent des informations externes pour définir les classes comme la taxinomie de WORDNET (Resnik, 1992), les catégories du thésaurus ROGET (Yarowsky, 1992), les catégories de sujets du dictionnaire LDOCE (Slator, 1992 ; Liddy & Paik, 1993) ou encore les ensembles conceptuels construits à partir des définitions de LDOCE (Luk, 1995). Ces modèles basés sur des classes surmontent partiellement le problème de la dispersion des données. Cependant, une partie de l'information est perdue par ces méthodes car l'hypothèse que tous les mots appartenant à une même classe ont un comportement identique est trop forte.

Les modèles basés sur une mesure de similarité (Dagan, Marcus & Markovitch, 1993 ; Dagan & Itai, 1994 ; Grishman & Sterling, 1993) exploitent la même idée de regroupement des informations sur des mots similaires, mais sans effectuer de stricts regroupements en classes. Chaque mot possède un ensemble potentiel distinct de mots similaires. Comme les modèles basés sur des classes, les modèles basés sur une mesure de similarité utilisent une métrique de similarité entre des cooccurrences. L'expérience de Dagan *et al.* (1993) semble montrer que les modèles basés sur une mesure de similarité obtiennent de meilleurs performances que les modèles basés sur des classes.

2.6 Approches mixtes

L'approche de Dahlgren (1988) présentée dans la section 2.3.3 constitue une approche mixte puisqu'elle utilise à la fois des phrases isolées, des informations syntaxiques et des raisonnements de sens commun.

L'approche de Yarowsky (1992) présentée dans la section 2.4.3 constitue également une approche mixte qui utilise le thésaurus ROGET et l'information extraite à partir de corpus pour générer des classes de mots dérivées des catégories de concepts du thésaurus ROGET. D'un autre côté, Resnik (1993a) et Agirre et Martinez (2001b) présentent une approche des contraintes de sélection semi-automatiquement extraite à partir de corpus.

McRoy (1992) décrit un système intégrant l'information de plusieurs bases de connaissances en s'appuyant, entre autres, sur les travaux de Hirst. Jusqu'à cette

époque, de nombreux petits systèmes ont été construits pour montrer ce qu'il faudrait faire. McRoy décrit un système s'attaquant à la désambiguïsation de tous les mots dans des textes non restreints. Le lexique utilisé comporte 13 000 sens et est composé d'un lexique général, toujours actif et contenant les sens généraux des mots, et de lexiques dépendants du domaine, activés uniquement quand le thème du texte correspond à un domaine particulier. Les sens des mots sont organisés en une hiérarchie conceptuelle. Le système commence par analyser la morphologie des mots et identifie les racines et affixes. Ensuite, leurs entrées dans les multiples lexiques sont identifiées et un étiquetage morphosyntaxique est réalisé. Pour prendre la décision finale, un système complexe de pondération considère les décisions prises en se basant sur des sources d'information distinctes comme l'étiquetage morphosyntaxique, la fréquence des lexies, les collocations, les contextes sémantiques, les restrictions de sélection et les indices syntaxiques. Les résultats obtenus sont encourageants.

Ng et Lee (1996) exploitent également plusieurs sources d'information comme l'étiquetage morphosyntaxique, la forme morphologique, les cooccurrences, les collocations et les relations syntaxiques *verbe–objet*. Toutes ces informations sont directement extraites du corpus.

Le *Cambridge Language Survey* (CLS) commercialise un étiqueteur sémantique (Harley & Glennon, 1997) qui utilise le dictionnaire électronique *Cambridge International Dictionary of English* (CIDE : Procter, 1995), construit en se basant sur un grand corpus de l'anglais constitué par le CLS. Cet étiqueteur consiste en quatre sous-étiqueteurs fonctionnant en parallèle dont la prise en compte des résultats permet de réaliser l'étiquetage final. Ces quatre sous-étiqueteurs sont respectivement basés sur :

- les collocations dérivées des exemples du dictionnaire électronique CIDE ;
- les catégories de sujets ;
- l'étiquetage morphosyntaxique ;
- les restrictions de sélection pour les verbes et les adjectifs.

Les résultats de ces sous-étiqueteurs sont combinés en utilisant un système de pondération analogue à celui de McRoy. Harley et Glennon annoncent une précision de désambiguïsation au niveau des *mots guides* du CIDE (comparable au niveau homographe de LDOCE) de 78%.

Dans la tradition du système de McRoy, Wilks et Stevenson (1997a, 1998) proposent un système combinant les informations suivantes :

- étiquettes morphosyntaxiques ;
- catégories de sujets issues du dictionnaire LDOCE ;
- définitions de dictionnaire.

Wilks et Stevenson annoncent une précision de désambiguïsation du niveau homographe de LDOCE de 88%.

Stevenson et Wilks (2001) présentent également un système combinant plusieurs sources d'information :

- filtrage basé sur l'étiquetage morphosyntaxique ;
- collocations générées à partir du corpus ;
- chevauchement avec les définitions du dictionnaire LDOCE ;
- catégories de sujets ;
- restrictions de sélection.

Stevenson et Wilks rapportent des résultats impressionnants : une précision de désambiguïsation au niveau homographe de LDOCE de 95% et au niveau des sens de 90%.

Mihalcea et Moldovan (1998) présentent une approche basée sur l'idée de *densité sémantique*. Ils utilisent la taxinomie et les définitions de WordNet en conjonction avec des statistiques issues directement de corpus récupérés sur Internet.

Les approches mixtes n'ont pas encore été massivement étudiées. Seuls quelques travaux y ont recours. Pourtant, c'est probablement de ce type d'approche qu'il faut attendre les meilleurs résultats.

2.7 Synthèse et conclusions

2.7.1 De la modélisation des connaissances ou du raisonnement à l'apprentissage automatique supervisé

La recherche en désambiguïsation lexicale possède maintenant une longue histoire qui débute avec celle de la traduction automatique. Très tôt, des approches basées sur la modélisation des connaissances ou du raisonnement voient le jour. Elles tirent parti des théories linguistiques et cherchent à modéliser le raisonnement humain. Ces approches peuvent être qualifiées de fortement typées intelligence artificielle. Elles utilisent des bases de connaissances spécifiques construites manuellement qui posent le problème difficilement surmontable de leur généralisation et de leur maintenance pour une couverture à grande échelle de la langue. Les approches utilisant des bases de connaissances prennent alors le relais. Ces approches montrent également leurs limites, principalement dues à l'inconsistance et à l'inadéquation pour le TAL des bases de connaissances utilisées. Pour ces raisons, les corpus deviennent ensuite la première source d'information. Mais les approches basées sur corpus sont confrontées au problème du manque de corpus lexicalement étiquetés et de la dispersion des données.

Une grande variété d'approches a donc été imaginée et investiguée, mais aucune d'entre elles n'est suffisamment performante pour être utilisée en situation réelle. Cependant, cette multitude d'approches fait ressortir une palette de sources d'information utiles pour la désambiguïsation lexicale. Agirre et Martinez (2001a) dressent ainsi une liste de ces sources d'information et tentent de les évaluer indépendamment.

2.7.2 Informations utiles pour la désambiguïsation lexicale

Voici un inventaire non exhaustif des sources d'information utiles et utilisées pour la désambiguïsation lexicale.

1. L'**étiquetage morphosyntaxique** permet de lever l'ambiguïté sur la catégorie grammaticale des vocables. Wilks et Stevenson (1997b), en utilisant uniquement l'étiquetage morphosyntaxique, atteignent une précision de désambiguïsation au niveau des homographes de 94%. Cependant, actuellement, un consensus semble émerger : l'étiquetage morphosyntaxique, et plus particulièrement la levée de l'ambiguïté sur la catégorie grammaticale des vocables, n'est pas du ressort de la désambiguïsation lexicale (Kilgarriff, 1997a ; Ng & Zelle, 1997). L'étiquetage morphosyntaxique est donc une phase indépendante et préalable à celle de la désambiguïsation lexicale. Cet étiquetage semble être un problème plutôt bien maîtrisé par la communauté du TAL. De nombreux étiqueteurs relativement fiables sont aujourd'hui disponibles pour un grand nombre de langues, et notamment pour le français [14]. L'étiquetage morphosyntaxique des mots du contexte fournit

14. Comme étiqueteur morphosyntaxique du français, nous pouvons par exemple citer le logiciel Cordial Analyseur développé par la société Synapse Développement (`http://www.synapse-fr.com/`) ou encore treetagger de Schmid (1994).

aussi des informations utiles à la désambiguïsation (Bruce & Wiebe, 1994b ; Ng & Lee, 1996, 2002 ; Yarowsky, 2000 ; Yarowsky *et al.*, 2001 ; Escudero *et al.*, 2000c, 2000b). Cet étiquetage permet également d'effectuer des généralisations (en réponse au problème de la dispersion des données) sur les mots du contexte du vocable à désambiguïser d'autant plus que la plupart des étiqueteurs effectuent également une lemmatisation. Cette source d'information est ainsi utilisée conjointement à celle des *collocations* (ou des *cooccurrences*) car elle permet de remplacer la forme morphologique brute des mots désignés par ces *collocations* (ou *cooccurrences*) par leur étiquette morphosyntaxique ou par leur lemme.

2. Les **collocations** sont des mots entretenant une relation particulière avec le mot à désambiguïser. Cette information est relativement difficile à acquérir et peut être partiellement capturée par des n-grammes contenant le mot à désambiguïser ainsi que par des relations syntaxiques binaires du type *nom–nom*, *adjectif–nom*, *verbe–objet*, *sujet–verbe*. Ng et Zelle (1997) définissent plus simplement les collocations comme des cooccurrences qui tiennent compte de l'ordre des mots.
3. Les **cooccurrences**, souvent définies en anglais par « *unordered set of surrounding words* » (ensemble non ordonné des mots du contexte), sont probablement actuellement la source la plus fiable et la plus utilisée dans le domaine de la désambiguïsation lexicale. Deux types d'informations véhiculées par les cooccurrences semblent se distinguer : les informations locales et les informations globales. Les informations locales correspondent à des dépendances locales comme l'adjacence au mot à désambiguïser, l'appartenance à un contexte réduit contenant ce mot. Les informations globales consistent en de larges fenêtres contenant 50 à 100 mots sous la forme de lemmes. Cette seconde information est plus rarement utilisée en raison du bruit qu'elle véhicule. Les cooccurrences peuvent utiliser l'information véhiculée par la source 1 et permettent de capturer partiellement l'information véhiculée par les sources 2, 5 et 6. La section 7.2.1 détaille comment cette information, ainsi que celle de la source 2, est utilisée dans quelques travaux de recherche en désambiguïsation lexicale.
4. Les organisations en **taxinomies** permettent de relier les différentes lexies d'un dictionnaire comme, par exemple, la taxinomie *IS-A* des noms du dictionnaire électronique WordNet.
5. Les **associations thématiques** permettent, par exemple, de relier des mots comme *batte* et *baseball*, mais aussi comme *garçon* (au sens garçon de café) et *table*. Les associations thématiques mettent en relation des mots qui ne sont pas forcément proches et pas forcément dans la même phrase. Cibler des mots éloignés par ce type de relation est bien plus pertinent que de considérer tous les mots contenus dans de larges fenêtres (de 50 à 100 mots par exemple).
6. Les **indices syntaxiques** peuvent également être utiles. Par exemple, le sens *prendre un repas* du verbe manger est intransitif tandis que ses autres sens (*avaler pour se nourrir*, *ronger*, *absorber*, *consommer*, etc.) sont transitifs.
7. Les **contraintes de sélection** permettent, par exemple, de préciser que le verbe *manger*, employé dans le sens *prendre un repas*, préfère un sujet de type *humain*.
8. L'**information sur le thème** du texte peut également s'avérer utile. Par exemple, Gale *et al.* (1992c) soutiennent l'idée qu'un vocable ne possède qu'un seul sens par discours. À l'opposé, Dahlgren (1988) observe que l'information sur le thème ne permet pas de lever l'ambiguïté de certains mots. Krovetz (1998) observe que des vocables peuvent avoir des occurrences multiples avec des sens différents au sein d'un même discours dans 33% des cas.

9. La **fréquence des sens** peut aussi s'avérer utile, notamment dans le cas où aucune autre source d'information ne permet de trancher.

La plupart des travaux réalisés dans le domaine de la désambiguïsation lexicale ne combinent pas toutes ces sources d'information, voire n'en utilisent qu'une seule. Quelques études (Dahlgren, 1988 ; McRoy, 1992 ; Ng & Lee, 1996 ; Harley & Glennon, 1997 ; Wilks & Stevenson, 1997a ; Stevenson & Wilks, 2001 ; etc.) tentent tout de même, et souvent avec succès, de combiner plusieurs sources d'information.

2.7.3 Tendance actuelle en désambiguïsation lexicale

Les expériences menées par Agirre et Martinez (2001a) confirment les observations de McRoy (1992) : les collocations (source 2, qui peuvent également être capturées par la source 3) ainsi que les associations thématiques (source 5) semblent être les sources d'information les plus efficaces pour la désambiguïsation lexicale. Ils remarquent également les bons résultats obtenus par des indices syntaxiques (source 6). D'un autre côté, l'utilisation de la taxinomie (source 4) ne donne pas de bons résultats et les contraintes de sélection (source 7) sont rarement applicables. Agirre et Martinez observent enfin que les résultats sont bien meilleurs lorsque les connaissances sont générées à partir de corpus manuellement étiquetés (cela est vrai pour les sources 2 ou 3, 5, 6, 7 et 9). L'impact des sources 1 et 8 n'a pas été évalué au cours de ces expériences.

Selon Ng (1997) un corpus manuellement lexicalement désambiguïsé de taille suffisante pour fournir une grande couverture de la langue (anglaise) n'est pas actuellement disponible. Depuis, des corpus manuellement lexicalement désambiguïsés commencent à voir le jour pour une langue comme l'anglais, notamment dans le cadre des campagnes d'évaluation SENSEVAL. Selon Ng, en l'état actuel des connaissances dans le domaine des techniques d'apprentissage supervisé et dans le domaine de la désambiguïsation lexicale, la disponibilité d'un corpus manuellement lexicalement désambiguïsé de taille importante permettrait d'aboutir à une désambiguïsation lexicale robuste, précise et de grande couverture. Toujours selon Ng, la limite maximale de l'effort à fournir pour la constitution d'un tel corpus est de l'ordre de 16 année-hommes. Ainsi, ce corpus pourrait être constitué en moins d'une année par une équipe de 16 annotateurs.

Il semble aujourd'hui clair que les approches supervisées [15] obtiennent de meilleurs résultats que les approches non supervisées sur des mots sélectionnés fortement polysémiques ainsi que sur des *pseudo-mots* artificiels (Ng, 1997 ; Escudero *et al.*, 2000a ; Kilgarrif & Rosenzweig, 2000 ; Agirre & Martinez, 2001a). De nombreuses études récentes sont d'ailleurs menées en utilisant de telles techniques (Pedersen *et al.*, 1997 ; Mooney, 1996 ; Ng, 1997 ; Escudero *et al.*, 2000c ; Yarowsky, 2000 ; Agirre & Martinez, 2000 ; Pedersen, 2002 ; Ng & Lee, 2002 ; etc.). De plus, cet engouement est renforcé par les campagnes d'évaluation SENSEVAL-1 et SENSEVAL-2 réalisées en 1998 et 2001, et par la campagne SENSEVAL-3 prévue pour 2004. Ces campagnes permettent l'émergence de standards dans le domaine de la désambiguïsation lexicale (Kilgarriff, 1997a, 1998a, 1998b ; Resnik & Yarowsky, 2000). Elles permettent également d'effectuer des comparaisons objectives entre les diverses équipes participantes ; or ces comparaisons étaient jusqu'alors très difficiles voire impossibles car chaque équipe travaille sur des corpus différents, des vocables différents, des distinctions de sens différentes, et utilise parfois des méthodes d'évaluation différentes. Un autre effet fortement positif de ces campagnes est l'émergence de corpus manuellement lexicalement désambiguïsés.

15. *i.e.* réalisant un apprentissage automatique sur des corpus manuellement lexicalement désambiguïsés.

Chapitre 3

Développement des outils principaux

3.1 Introduction

Très tôt, dans le cadre de ce travail de thèse, des besoins en outils adaptés se sont fait ressentir. Ces besoins ne se limitent pas au cadre strict de ce travail puisqu'ils apparaissent en amont, pour la constitution et l'étiquetage du corpus, et s'étendent au niveau de l'enseignement et de la recherche au sein de notre équipe. L'effort pour répondre partiellement à ces besoins constitue une part importante (probablement la plus importante en terme d'effort) de ce travail de thèse.

Dans la **section 3.2** nous exprimons nos besoins en applications informatiques.

Dans la **section 3.3**, nous présentons les outils existants vers lesquels nous aurions pu choisir de nous tourner. Nous expliquons également pourquoi nous n'avons pas fait ce choix.

Dans la **section 3.4** nous présentons une bibliothèque qui permet de modéliser des requêtes complexes à appliquer sur des corpus. Deux applications ont été construites autour de cette bibliothèque, (Win/DOS)LoX et CooLoX.

L'application (Win/DOS)LoX est un outil puissant de recherche et d'observation de phénomènes linguistiques dans les corpus écrits, qui permet de formater librement le résultat des requêtes. L'application CooLoX est un puissant concordancier. Ces deux applications sont brièvement décrites dans la **section 3.5**. Leur manuel d'utilisation complet figure en annexes A et B.

3.2 Expression des besoins

3.2.1 Présentation des besoins

Les chercheurs en linguistique, voire plus généralement en TAL, ont souvent besoin d'un concordancier puissant supportant des requêtes complexes et permettant de travailler sur des corpus morphosyntaxiquement désambiguïsés.

Le travail effectué par l'équipe de recherche DELIC dans le cadre du projet SyntSem nécessite un outil qui permet et facilite l'étiquetage vertical (Reymond, 2001 ; voir également section 4.4.1 pour une description sommaire de cette technique).

Dans le cadre de notre étude, un outil capable de repérer des phénomènes linguistiques en cooccurrence avec un mot donné est indispensable. Cet outil doit être capable de générer les données d'apprentissage nécessaires à la mise en œuvre de techniques d'apprentissage supervisé pour aboutir à une désambiguïsation lexicale automatique. Les traitements que nous avons à effectuer sont nombreux et doivent être automatisés par l'intermédiaire de scripts. Il faut donc que cet outil supporte le travail orienté *ligne de commande*.

Notre besoin est donc triple :

1. un concordancier évolué ;
2. un outil permettant de réaliser de l'étiquetage vertical ;
3. un outil puissant de recherche de phénomènes linguistiques autorisant le formatage libre du résultat des requêtes et acceptant d'être utilisé par l'intermédiaire de scripts de commande.

Il faut que ces outils puissent s'adapter à nos besoins susceptibles d'évoluer en fonction des directions de recherche et des informations disponibles (corpus étiquetés, bases de données, corpus arborés, corpus oraux avec transcription, etc.).

3.2.2 Description des outils souhaités

Le concordancier

Un concordancier est un outil capable d'effectuer des recherches dans un texte et de présenter le résultat de ces recherches « en contexte ». De manière classique, la présentation se fait sur trois colonnes. Dans la colonne du milieu se trouve le résultat de la recherche. Dans la colonne de gauche se trouve le contexte gauche du résultat de la recherche, et dans la colonne de droite le contexte droit. Un formalisme de requête permet d'énoncer ce que nous désirons voir dans la colonne du milieu. Ce formalisme peut être éventuellement utilisé pour filtrer le résultat en fonction du contexte gauche ou du contexte droit. Les fonctionnalités les plus classiques d'un tel concordancier sont :

- une possibilité de tri sur chacune des trois colonnes ;
- la spécification de la taille des contextes ;
- la possibilité de voir ponctuellement un résultat dans un contexte élargi ;
- une possibilité d'exportation des concordances, sous forme de fichiers textes par exemple.

Le concordancier doit posséder une interface graphique pour faciliter et rendre plus interactive son utilisation. Il doit fonctionner sous *Microsoft Windows* puisque c'est l'environnement que nous utilisons actuellement principalement, que ce soit pour l'enseignement ou la recherche. Il doit enfin supporter des requêtes complexes et être capable de travailler sur des corpus lemmatisés et morphosyntaxiquement étiquetés. En effet, la lemmatisation et l'étiquetage morphosyntaxique permettent d'écrire des requêtes précises et parfois complexes très simplement. Par exemple, en se servant de la lemmatisation, la requête sur un vocable donné est triviale alors qu'elle est souvent impossible (en raison des ambiguïtés catégorielles) et approximée de manière complexe et fastidieuse (pour couvrir toutes les formes fléchies d'un verbe par exemple) lorsque la lemmatisation n'est pas disponible.

L'étiquetage vertical

La fonction d'étiquetage dont nous avons besoin peut tout à fait s'insérer dans un concordancier puisqu'il s'agit d'un étiquetage vertical. Concrètement, il suffit d'ajouter

une colonne aux trois colonnes déjà présentes dans le concordancier. Cette colonne reçoit les étiquettes qui peuvent être saisies manuellement dans chaque case, ou être affectées par lots ou par sélections multiples. Ces étiquettes doivent, bien-entendu, être ensuite injectées dans le corpus source.

L'outil de recherche de phénomènes linguistiques

Cet outil doit permettre d'écrire des requêtes complexes sur des corpus étiquetés et permettre de formater librement le résultat de ces requêtes. Contrairement à certains logiciels qui ne permettent que l'extraction de concordances au format relativement figé, le formatage libre du résultat des requêtes permet leur réutilisation par des programmes ultérieurs et autorise une grande diversité d'applications, s'écartant largement du cadre des simples concordanciers. Comme nous l'avons précisé dans l'introduction, cet outil doit pouvoir être utilisé par l'intermédiaire de scripts de commande.

3.2.3 Caractéristique commune à nos besoins

Au cœur de tous les besoins énoncés se trouve un *langage de requête*. Cette caractéristique commune à tous ces besoins est également la plus difficile à réaliser d'un point de vue algorithmique. L'idéal serait de disposer d'une bibliothèque qui implémente ce langage. Cette bibliothèque faciliterait grandement le développement d'applications, répondant à des besoins bien particuliers, qui nécessitent ce type de langage de requête, puisque la partie critique du programme serait déjà écrite. Cette bibliothèque devrait disposer d'un mécanisme d'abstraction du corpus pour permettre de faire facilement évoluer les applications développées en fonction des sources d'information disponibles (*i.e.* elle devrait pouvoir travailler virtuellement sur n'importe quel type de corpus et se connecter à n'importe quelle base de données). De plus, elle devrait être écrite de manière à être indépendante de l'environnement. Ainsi, elle permettrait de développer des applications graphiques, ou orientées *ligne de commande*, aussi bien pour des environnements *Microsoft Windows* que *Macintosh* ou *Unix/Linux*.

Le formalisme des requêtes modélisé par cette bibliothèque devrait permettre d'écrire facilement des requêtes simples. En effet, des linguistes non familiarisés avec le monde de l'informatique, ainsi que des étudiants venant de divers horizons, seront amenés à l'utiliser. Le formalisme devrait également permettre d'écrire des requêtes complexes pour les chercheurs les plus exigeants. De plus, dans le souci de faciliter sa prise en main, il serait judicieux que ce formalisme se rapproche de mécanismes existants et largement employés par la plupart des chercheurs en linguistique informatique (comme le formalisme des expressions régulières par exemple).

3.3 Outils existants

Dans cette section, nous présentons les principaux outils existants qui pourraient éventuellement répondre à certains de nos besoins. Nous donnons pour chacun un aperçu sommaire de leurs fonctionnalités. Cette liste d'outils n'est pas exhaustive car certains outils nous ont peut-être échappés ou sont trop récents pour avoir été pris en considération (comme par exemple UNITEX [1]).

1. UNITEX est un logiciel gratuit développé par Sébastien Paumier à l'Institut Gaspard-Monge (IGM), à l'Université de Marne-la-Vallée (France). Les fonctionnalités d'UNITEX sont proches de celles d'INTEX que nous présentons section 3.3.10. Site Internet : `http://www-igm.univ-mlv.fr/unitex/` .

3.3.1 British National Corpus : SARA

Le corpus national anglais (BNC [2]) est une collection de 100 millions de mots extraits de discours écrits et oraux de sources très variées. Ce corpus a été constitué pour représenter de manière significative l'anglais contemporain aussi bien parlé qu'écrit.

Un outil de recherche a été développé spécifiquement pour ce corpus. Cette application, appelée SARA (SGML Aware Retrieval Application) fonctionne sous l'environnement *Microsoft Windows*. Elle permet aux non-experts de rechercher rapidement dans le corpus BNC des exemples spécifiques de mots, de phrases, ou de motifs de mots. Le résultat peut être trié et affiché de différentes manières. Les recherches peuvent porter sur des contextes SGML particuliers (titres de journaux, citations, etc.) ou sur des types de textes particuliers (écrits scientifiques, discours d'une femme d'âge moyen du sud de l'Angleterre, etc.). Les recherches sur les lemmes et les étiquettes morphosyntaxiques en utilisant des expressions régulières sont également possibles.

SARA peut être obtenu gratuitement par tout utilisateur possédant une licence pour le corpus BNC, ou enregistré auprès du service en ligne du corpus BNC.

3.3.2 CobuildDirect

CobuildDirect [3] est un service en ligne qui permet de rechercher des concordances et des cooccurrences dans un corpus de 56 millions de mots d'anglais moderne écrit [4] et oral [5]. Le corpus est morphosyntaxiquement étiqueté et lemmatisé.

Le langage de requête permet de décrire un ensemble de mots (*i.e.* une liste possible de mots) en utilisant des formules logiques sur des expressions régulières et des étiquettes morphosyntaxiques. Un séparateur permet de concaténer des expressions logiques pour décrire des suites d'ensembles de mots (donc des portions de texte). Par exemple, l'expression :

```
bridge/(NOUN|VERB)+!on
```

décrit les occurrences du mot *bridge* utilisé comme un nom ou un verbe et non suivi par *on*.

L'utilisation de CobuildDirect est soumise à licence.

Un CD-ROM sur les collocations est également disponible. Il recense 140 000 collocations à travers 2,6 millions d'exemples réels issus d'un corpus contenant 200 millions de mots. Les mots peuvent être vus en contexte grâce à une interface de type concordancier.

3.3.3 Conc

Conc [6] est un concordancier pour *Macintosh*. Il est composé de trois fenêtres, la première contient le corpus, la seconde le concordancier, la dernière l'index. Les trois fenêtres sont interactives et synchronisées.

2. Adresse : British National Corpus ; Oxford University Computing Services ; 13 Banbury Road ; Oxford OX2 6NN. Site Internet : `http://info.ox.ac.uk/bnc/index.html` . Courrier électronique : `natcorp@oucs.ox.ac.uk` .

3. Site Internet : `http://titania.cobuild.collins.co.uk/index.html` . Courrier électronique : `direct@cobuild.collins.co.uk` .

4. La partie écrite est très variée et comporte des livres de fiction et de non fiction, aussi bien anglais qu'américains, des journaux, des revues, des courriers personnels, des publicités, des brochures, etc.

5. La partie orale correspond à des discours de la vie courante et de situations informelles. Sa taille est de 10 millions de mots.

6. Site Internet : `http://www.sil.org/computing/conc/` . Courrier électronique : `WWW@sil.org` .

Les mots ciblés peuvent être définis par des expressions régulières similaires à celles de la commande GREP. Les corpus peuvent être de simples corpus bruts ou des corpus annotés.

3.3.4 CONCORDANCE

CONCORDANCE [7] est un concordancier convivial fonctionnant sous *Microsoft Windows*. Il permet en outre d'obtenir des informations sur les cooccurrences et comporte un lemmatiseur intégré. Il travaille avec des corpus de texte brut ou avec balisage léger. *Concordance* n'utilise pas directement le corpus d'origine mais génère son propre corpus de travail. La taille de ce dernier est bien supérieure à celle du corpus original, ce qui peut devenir extrêmement pénalisant sur de gros corpus. Le formalisme de requête est simple et rudimentaire.

3.3.5 MONOCONC PRO

MONOCONC PRO [8] est un concordancier fonctionnant sous *Microsoft Windows*. Il est à rapprocher de CONCORDANCE. Son interface est légèrement moins conviviale mais il supporte des corpus étiquetés, et son formalisme de requête est plus puissant. En outre, MONOCONC PRO travaille directement sur le corpus et ne génère pas de fichier intermédiaire.

3.3.6 WORDSMITH TOOLS

WORDSMITH TOOLS [9] est une collection d'outils destinés à assister les lexicographes et les linguistes dans leur travail sur corpus. WORDSMITH TOOLS comporte les applications suivantes :

WORDSMITH TOOLS est le panneau de contrôle des différentes applications ;

CONCORD est un concordancier qui permet également d'obtenir des informations sur les collocations et d'effectuer quelques traitement statistiques ;

WORDLIST permet de générer des listes de mots et de les comparer ;

KEYWORDS permet de rechercher des mots clefs à partir de listes de mots ;

SPLITTER est un utilitaire pour diviser des textes trop volumineux ;

TEXT CONVERTER permet d'effectuer des opérations de recherche et de remplacement de chaînes de caractères ;

VIEWER est un utilitaire d'affichage de texte.

Le concordancier CONCORD est capable de travailler sur des corpus étiquetés. Son formalisme de requête est basé sur les expressions régulières.

3.3.7 TGREP

TGREP [10] a été développé par Richard Pito à l'institut Benjamin Franklin.

7. CONCORDANCE Copyright ©1999, 2000 R.J.C. WATT. Site Internet : `http://www.rjcw.freeserve.co.uk/` . Courrier électronique : `R.J.C.Watt@dundee.ac.uk` .

8. MONOCONC PRO Copyright ©1996, 2000 Michael Barlow. Adresse : Athelstan ; 2476 Bolsover, Suite 464 ; Houston, TX 77005 USA. Site internet : `http://www.athel.com/rade.html` . Courrier électronique : `info@athel.com` ou `barlow@athel.com` .

9. WORDSMITH TOOLS Copyright ©1998 Mike Scott. Site Internet : `http://www.liv.ac.uk/ms2928/wordsmith/index.htm` . Courrier électronique : `Mike.Scott@liv.ac.uk` .

10. Copyright ©1993, 1994 Richard Pito. Site internet : `http://mccawley.cogsci.uiuc.edu/corpora/tgrep.pdf` . Courrier électronique : `pito@unagi.cis.upenn.edu` .

Introduction

TGREP est un outil permettant la recherche et l'extraction de structures particulières dans des corpus arborés. Il s'agit en fait d'une extension de la fonction GREP, bien connue des systèmes d'exploitation *Unix* ou *Linux*, pour les arbres. Les expressions que supporte TGREP désignent des nœuds et des relations entre ces nœuds. L'expression est alors appliquée à l'ensemble du corpus et tous les arbres décrits par cette expression sont sélectionnés pour être affichés. TGREP offre plusieurs façons de formater ce que l'utilisateur désire afficher à partir des arbres sélectionnés. TGREP est conçu pour être très rapide mais nécessite des textes précompilés.

TGREP fonctionne sur des corpus arborés, en particulier sur le corpus Penn Treebank (Marcus *et al.*, 1994 ; Marcus & Marcinkiewicz, 1993) dont voici un extrait à titre d'illustration :

```
(TOP (S (NP-SBJ my best friend)
        (VP gave
                (NP me)
                (NP chocolate)
                (NP yesterday))
        .))
```

Syntaxe des nœuds

La spécification d'un nœud se fait soit de manière explicite (par exemple `NP`, `S`, `chocolate`), soit en utilisant une expression régulière (par exemple `/.*day/`), soit en utilisant le nœud spécial qui décrit n'importe quel nœud (*i.e.* __). TGREP supporte également un opérateur de disjonction, `|`, entre les nœuds. Par exemple, l'expression

```
/^[Cc]hild.*$/|kid|youngster
```

décrit des nœuds comme : *child*, *Child*, *children*, *Children*, *kid* et *youngster*.

Syntaxe des relations entre les nœuds

Voici la syntaxe des relations possibles entre les nœuds :

`A < B`	`A` est le père de `B` ;
`A > B`	`A` est un fils de `B` ;
`A <x B`	`B` est le `x`^{e} fils de `A` ;
`A <- B`	`B` est le dernier fils de `A` ;
`A <-x B`	`B` est le `x`^{e} fils de `A` en partant du dernier ;
`A << B`	`A` est un ascendant de `B` ;
`A <> B`	`A` est un descendant de `B` ;
`A <<, B`	`B` est le fils le plus à gauche de `A` ;
`A <<' B`	`B` est le fils le plus à droite de `A` ;
`A . B`	`A` précède immédiatement `B` ;
`A .. B`	`A` précède `B` ;
`A $ B`	`A` et `B` sont des frères ;
`A $. B`	`A` et `B` sont des frères et `A` précède immédiatement `B` ;
`A $.. B`	`A` et `B` sont des frères et `A` précède `B`.

La négation de ces relations existe. Il suffit d'adjoindre le symbole `!` devant les symboles de relation pour les obtenir.

Écriture des expressions

Les expressions sont formées de nœuds et de relations entre ces nœuds. Par exemple, l'expression `S < NP << S` décrit un nœud `S` père d'un nœud `NP` et ascendant d'un nœud `S`. La seconde relation `<< S` se rapporte donc au premier nœud `S` et non au nœud `NP`. Cette syntaxe est choisie pour se passer de l'opérateur de conjonction.

L'utilisation des parenthèses est, bien entendu, autorisée. Par exemple, l'expression `S < (NP << S)` décrit un nœud `S` père d'un nœud `NP` étant lui-même ascendant d'un nœud `S`.

Formatage du résultat

Par défaut, TGREP retourne comme résultat le sous-arbre décrit par l'expression avec tous les descendants. Il est possible de ne retourner que certains nœuds ; pour cela, les nœuds en question doivent être précédés d'une quote (`'`) dans l'expression.

Enfin, des commutateurs supplémentaires passés par la *ligne de commande* permettent de retourner toute la ligne (`-w`), de retourner le nom du fichier (`-f`) et les numéros de ligne (`-l`), ou encore de retourner des phrases du contexte (`-c X`).

3.3.8 TGREP2

Introduction

TGREP2 [11] est une refonte complète de TGREP offrant un grand nombre de nouvelles fonctionnalités, tout en restant compatible de manière ascendante avec pratiquement toutes les commandes de TGREP. TGREP2 est développé par Douglas Rohde.

Améliorations par rapport à TGREP

Voici les améliorations majeures apportées à TGREP2 par rapport à TGREP :

- un certain nombre de relations entre les nœuds a été ajouté aux relations déjà existantes dans TGREP ;
- les opérations booléennes entre les relations sont supportées par TGREP2 ;
- les nœuds peuvent être nommés pour y faire référence dans les expressions, ou pour désigner les portions dont l'utilisateur désire afficher le résultat ; cette technique permet également de générer des graphes qui n'ont pas une structure d'arbre;
- les options du formatage du résultat sont plus riches.

3.3.9 IMS Corpus Workbench

Introduction

Développé à l'institut de traitement automatique des langues de l'université de Stuttgart en 1994, l'IMS CORPUS WORKBENCH [12] est un environnement intégré et extensible de requêtes sur des corpus fonctionnant sous un environnement de type *Unix/Linux*. L'IMS CORPUS WORKBENCH est présenté par Christ (1994).

11. Copyright ©2001, Douglas L. T. Rohde. TGREP2 est disponible gratuitement pour les étudiants, les enseignants et toute organisation à but non lucratif. Site Internet : `http://www.cs.cmu.edu/ dr/Tgrep2/` . Courrier électronique : `dr+tg@cs.cmu.edu` .

12. Copyright ©2001, Université de Stuttgart ; Institut du traitement automatique des langues ; Azenbergstr ; 12, 70174 Stuttgart, Alemagne. Site Internet : `http://www.ims.uni-stuttgart.de/CorpusWorkbench/` .

Le langage de requête supporte un nombre quelconque d'attributs pour chaque mot du corpus. Les requêtes sont des expressions régulières sur les mots du corpus (et non pas sur les caractères comme pour les expressions régulières classiques). Il est également possible de se connecter à des bases de données externes, comme un thésaurus, et de travailler sur des textes parallèles.

Les résultats affichés sont les mots trouvés en contexte. Le résultat peut aussi être retourné au format HTML ou LaTeX.

Le langage de requête est interprété par CQP (*Corpus Query Processor*). Cela présuppose que les corpus soient enregistrés auprès de CQP et soient encodés d'une manière spécifique.

Syntaxe des requêtes

Une requête portant sur un mot isolé est une expression logique sur les attributs du mot. Par exemple, la requête `[pos="DT"]` décrit tous les mots dont l'attribut `pos` (*i.e.* l'étiquette morphosyntaxique) est `DT` (*i.e.* déterminant). La chaîne de caractères peut également être remplacée par une expression régulière, par exemple `[pos="N.*"]`. Dans cette expression, `N.*` désigne toutes les chaînes commençant par `N`. Lorsque l'attribut est le mot lui-même, l'expression peut être abrégée ; par exemple `[word="Clinton"]` peut s'écrire `"Clinton"`. L'expression

```
[word="rain" & pos="NN"]
```

décrit toute les occurrences du mot *rain* où ce mot est un nom.

Une requête portant sur des séquences de mots se fait en juxtaposant des requêtes sur des mots isolés. Comme pour les expressions régulières, les multiplicateurs sont autorisés. Par exemple, la requête

```
"give" [pos!="SENT"]{0,4} "up"
```

décrit des séquences de mots commençant par le mot *give* suivi de zéro à quatre mots dont l'étiquette morphosyntaxique est différente de `SENT` suivi du mot *up*.

3.3.10 INTEX

Introduction

Développé par Max Silberztein au LADL (Laboratoire d'Automatique Documentaire et Linguistique), INTEX [13] est un environnement de développement utilisé pour construire des descriptions formalisées à large couverture des langues naturelles, et les appliquer à des textes de taille importante en temps réel. Les descriptions des langues naturelles sont formalisées sous la forme de dictionnaires électroniques, de grammaires représentées par des graphes à états finis et de lexiques-grammaires.

INTEX est aussi utilisé pour le traitement de corpus : indexation de motifs morphosyntaxiques, d'expressions figées ou semi-figées, de concordances lemmatisées, et étude statistique des résultats.

INTEX fonctionne sous un environnement *Microsoft Windows* et est très exigeant en terme de puissance de calcul et d'occupation mémoire.

13. Copyright ©ASSTRIL 1999-2000. LADL, 5 Bd Descartes, CHAMPS-SUR-MARNE ; F-77454 MARNE-LA-VALLEE CEDEX 2. Courrier électronique : `mmathis@univ-mlv.fr` . Site Internet : `http://ladl.univ-mlv.fr/Intex/index.html` .

Description sommaire

Une caractéristique essentielle d'INTEX est que tous les objets traités (textes, dictionnaires, grammaires, requêtes) sont à un moment ou à un autre représentés par des transducteurs à états finis. Un transducteur à état fini est un graphe qui représente un ensemble de séquences en entrée, et leur associe des séquences produites en sortie. Toutes les opérations effectuées par INTEX se ramènent à un ensemble limité d'opérations sur des transducteurs.

Les expressions régulières (analogues au langage de requête décrit dans la section 3.3.9) constituent un moyen rapide pour spécifier des automates finis simples, sans avoir à dessiner de graphes. INTEX supporte de telles expressions. En revanche, dès que la structure à rechercher se complique, il est préférable de construire des graphes représentant des automates finis, cas particulier des transducteurs à états finis.

Les fonctionnalités disponibles grâce à l'interface graphique d'INTEX sont aussi disponibles par la *ligne de commande*. Il est donc possible d'appeler ces commandes depuis un programme écrit en PERL, C++, JAVA, etc.

À ce jour, plus d'une centaine d'utilisateurs utilisent INTEX comme outil de recherche ou d'enseignement dans une dizaine de pays.

3.3.11 Pourquoi ces outils ne nous suffisent-ils pas ?

L'outil de recherche SARA ne pourrait répondre qu'au besoin de concordancier, mais pas au besoin d'étiquetage vertical ni au besoin d'un outil de recherche de phénomènes linguistiques autorisant le formatage libre du résultat des requêtes et acceptant d'être utilisé par l'intermédiaire de scripts de commande. De plus, ce logiciel est dédié au corpus BNC (corpus anglais).

COBUILDDIRECT est un service en ligne permettant d'accéder à un corpus anglais. Il ne correspond donc pas du tout à notre attente. De plus, le langage de requête possède quelques lacunes et le formatage des résultats est relativement figé et de type concordancier.

CONC est un simple concordancier ne fonctionnant que sous *Macintosh*.

CONCORDANCE est un bon concordancier extrêmement convivial. Son formalisme de requête est cependant assez pauvre. Il ne supporte pas de corpus étiqueté et ne peut se connecter à une base de données externe.

MONOCONC PRO est un très bon concordancier. Il supporte les corpus étiquetés mais ne peut se connecter à une base de données externe. Ces trois concordanciers n'offrent aucun module d'étiquetage. Ils ne permettent pas de formater librement le résultat des requêtes pour une utilisation autre que celle du type concordancier. Ces outils ne correspondent donc pas à notre attente.

La collection d'outils WORDSMITH TOOLS ne correspond pas à notre attente pour les mêmes raisons que les deux concordanciers précédents.

TGREP travaille sur des corpus arborés. Or, il n'existe actuellement pas de logiciel capable de produire ces corpus de manière automatique et fiable. TGREP n'est donc utilisable que sur un nombre très restreint de corpus. TGREP ne peut travailler avec des corpus étiquetés (des corpus où chaque mot possède une étiquette morphosyntaxique, une lemmatisation, etc.) ni avec des bases de connaissances. Le formatage du résultat avec TGREP est extrêmement pauvre et ne correspond pas du tout à notre attente. Aucune bibliothèque n'est disponible pour réaliser des applications particulières reposant sur le même mécanisme de requête. TGREP est orienté *ligne de commande*. Ne possédant pas d'interface, son utilisation devient vite fastidieuse. Il n'est donc pas adéquat pour une utilisation de type concordancier et ne permet pas l'étiquetage de corpus. Enfin, cet

outil est difficilement utilisable pour des séances de travaux pratiques destinées à des étudiants sous *Microsoft Windows*. Quant à TGREP2, sa version 1.0 n'est disponible que depuis le 22 Mai 2001, date postérieure à la réalisation des outils présentés dans ce mémoire. Pour l'utilisation que nous voudrions en faire, TGREP2 possède la plupart des défauts de TGREP.

Le langage de requête de l'IMS CORPUS WORKBENCH est riche et pourrait pratiquement satisfaire nos besoins. Cependant, le formatage des résultats est bien trop pauvre. Le concordancier associé (XKWIC) n'est pas très évolué et ne permet aucun étiquetage. De plus, aucune bibliothèque n'est disponible pour développer ses propres applications basées sur ce langage de requête. Enfin, la mise en œuvre de l'IMS CORPUS WORKBENCH n'est pas simple, est soumise à licence et ne s'accorde pas avec le matériel dont nous disposons.

INTEX est un outil extrêmement puissant. Il couvre la plupart de nos besoins. Cependant, INTEX ne supporte pas directement les corpus annotés. Il n'est donc pas possible de travailler avec des corpus morphosyntaxiquement désambiguïsés. De plus, le concordancier d'INTEX ne permet pas de réaliser un étiquetage manuel. Bien qu'il soit possible de générer des commandes INTEX dans des programmes, aucune bibliothèque n'est fournie à ce jour. L'intégration d'INTEX dans un programme est donc forcément très sommaire et n'offre pas un grand intérêt dans le cadre d'une application avec une interface graphique amenée à interagir fortement avec l'utilisateur. Enfin, INTEX est trop exigeant en terme de matériel et surtout trop complexe pour être utilisé par des étudiants et des linguistes peu familiarisés aux outils informatiques et mathématiques. L'utilisation d'INTEX est également soumise à licence.

Certains des concordanciers présentés répondent probablement en partie à nos besoins. INTEX répond peut être à notre besoin d'outil puissant de recherche de phénomènes linguistiques. Cependant, nos besoins ne sont pas ponctuels mais importants, précis et susceptibles d'évoluer dans le temps. Il nous paraît difficile de nous accommoder d'une application répondant approximativement à nos besoins. Une telle approche serait rentable à court terme mais préjudiciable à long terme. Pour toutes ces raisons, il nous paraît justifié de développer nos propres outils. Pour cela, nous choisissons de développer une bibliothèque qui permet de modéliser des requêtes complexes. Cette bibliothèque sera utilisée et réutilisable pour de nombreuses applications nécessitant un langage de requête. La polyvalence et le caractère réutilisable de cette bibliothèque justifie également l'effort nécessaire à fournir pour son développement. En choisissant cette politique, nous sommes également certain de pouvoir faire évoluer nos outils en fonction de nos besoins.

3.4 La bibliothèque LoX

3.4.1 Introduction

Nous décrivons dans cette section une bibliothèque, développée en `C++`, qui permet de modéliser des requêtes complexes à appliquer sur des corpus. Il ne s'agit pas d'une application à part entière, mais d'une boîte à outils regroupant un ensemble de fonctions permettant de modéliser un langage de requête, destinée au développement d'applications d'exploration et d'exploitation de corpus écrits.

Les linguistes, habitués au travail sur les corpus écrits, sont généralement familiers avec les expressions régulières. Ces expressions se situent au niveau des caractères et permettent de décrire de manière formelle des classes de chaînes de caractères. Notre bibliothèque LoX permet de modéliser des méta-expressions régulières (dont le for-

malisme se rapproche de celui de IMS Corpus Workbench décrit section 3.3.9). Ces méta-expressions se situent au niveau des mots et permettent de décrire de manière formelle des classes de suites de mots, donc des portions de texte.

La bibliothèque LoX reconnaît actuellement deux types de corpus, les corpus de texte brut (format le plus classique et le plus simple), et les corpus étiquetés au format tabulaire (comme ceux générés par le logiciel Cordial Analyseur). En travaillant sur des corpus étiquetés, il est possible de se référer indifféremment aux mots ou aux étiquettes des mots (comme le lemme ou l'étiquette morphosyntaxique).

3.4.2 Définitions préliminaires

Par la suite, nous utiliserons régulièrement le terme de *correspondance*. De manière simplifiée, une correspondance désigne une portion de corpus décrite par une requête. Les correspondances sont en fait le résultat *brut* [14] de l'application d'une requête sur le corpus.

Prenons, par exemple, l'extrait de corpus suivant :

> « **Ce projet** risque de provoquer **une catastrophe** écologique sur **une surface** boisée équivalente à celle de la France. »

Dans cet extrait, la requête *déterminant suivi d'un nom commun* retourne trois correspondances désignant les trois portions en gras.

Définition 3.1 – Correspondance –
Une requête permet de décrire de manière formelle un phénomène linguistique. Une instance de ce phénomène dans un corpus, c'est-à-dire une portion de corpus décrite par la requête, est appelée une correspondance. À une correspondance sont associées un certain nombre de références (cf. définition 3.4) qui permettent de désigner certains mots de la correspondance.

Les références peuvent être définies par l'utilisateur lors de l'écriture de la requête. Notons qu'il existe aussi des références prédéfinies permettant de se référer au premier mot et au dernier mot de la correspondance.

Pour des raisons de complétude, lors de l'application d'une requête, toutes les correspondances possibles sont retournées. Une même requête peut très bien décrire plusieurs fois (*i.e.* de plusieurs manières) la même portion de corpus. Ainsi, l'ensemble des correspondances d'une requête peut très bien comporter plusieurs correspondances qui désignent la même portion de corpus. Nous appelons de telles correspondances des *correspondances similaires*.

Définition 3.2 – Correspondance similaire –
Nous qualifions deux correspondances de correspondances similaires quand elles désignent la même portion de corpus.

Prenons par exemple la requête suivante :

$$\underbrace{\texttt{[]\{0,2\}}}_{expr_1}\underbrace{\texttt{[]\{0,2\}}}_{expr_2}$$

Chacune de ces deux expressions ($expr_1$ et $expr_2$) décrit zéro à deux mots quelconques. La requête complète décrit donc n'importe quelle portion de corpus contenant zéro à quatre mots. En considérant deux mots juxtaposés, mot_A et mot_B, dans un corpus

14. Il s'agit du résultat de l'application de l'objet `C++` *requête* sur le corpus. C'est au concepteur du logiciel utilisant la bibliothèque LoX de choisir la façon dont ces correspondances sont utilisées.

quelconque, parmi toutes les portions décrites, nous trouvons trois correspondances similaires comportant les deux mots mot_A et mot_B :

$$\underbrace{mot_A}_{expr_1}\underbrace{mot_B}_{expr_1} \qquad \underbrace{mot_A}_{expr_1}\underbrace{mot_B}_{expr_2} \qquad \underbrace{mot_A}_{expr_2}\underbrace{mot_B}_{expr_2}$$

Dans la première correspondance, les deux mots sont décrits par l'expression $expr_1$. Dans la seconde, le premier mot est décrit par l'expression $expr_1$ tandis que le second est décrit par l'expression $expr_2$. Dans la dernière, les deux mots sont décrits par l'expression $expr_2$.

3.4.3 Présentation informelle des différents formalismes

Qu'est-ce qu'une expression régulière ?

La bibliothèque LoX étant construite autour des expressions régulières et de leur généralisation, son utilisation nécessite de bien se familiariser avec le langage des expressions régulières.

Concrètement, le formalisme des expressions régulières permet de décrire des ensembles de chaînes de caractères. Étant donné une expression régulière et une chaîne de caractères, deux cas de figure se présentent :

- l'expression régulière décrit la chaîne de caractères ;
- l'expression régulière ne décrit pas la chaîne de caractères.

Si l'expression régulière décrit la chaîne de caractères, c'est que cette chaîne appartient à l'ensemble des chaînes de caractères décrites par l'expression régulière.

Dans ce qui suit, nous illustrons le fonctionnement des expressions régulières en les appliquant à l'ensemble des chaînes de caractères *C1={abréviation, abrite, absence, absentait, absolue, abstenir, abstenu, absurde, absurdement}*. Nous n'abordons pas tous les aspects des expressions régulières, mais seulement les plus simples et les plus utiles. Pour une description plus complète et plus formelle des expressions régulières, se référer à la section 3.4.7.

Expression régulière simple

Par convention, nous écrivons toujours les expressions régulières, ainsi que les chaînes de caractères, entre guillemets (`"`). L'expression régulière `"en"` décrit toutes les chaînes contenant comme sous-chaîne la chaîne *en*, soit les chaînes *abs**en**ce, abs**en**tait, abst**en**ir, abst**en**u, absurdem**en**t* de l'ensemble $C1$.

Les méta-caractères sont des caractères spéciaux qui peuvent être utilisés au sein d'une expression régulière. Ils ont un pouvoir expressif plus grand qu'un simple caractère.

Méta-caractère point (`.`) des expressions régulières

Le méta-caractère point (`.`) est un caractère décrivant n'importe quel caractère. Ainsi, l'expression régulière `"s...e"` décrit toutes les chaînes contenant comme sous-chaîne une chaîne commençant par le caractère **s** suivi de trois caractères quelconques suivi du caractère **e**. Donc `"s...e"` décrit les chaînes *ab**solue**, ab**surde**ment* de l'ensemble $C1$.

word
La
construction
de
centaines
de
digues
et
barrages
entraînera

jeton	lemme	ems
La	le	DETDFS
construction	construction	NCFS
de	de	PREP
centaines	centaine	NCFP
de	de	PREP
digues	digue	NCFP
et	et	COO
barrages	barrage	NCMP
entraînera	entraîner	VINDF3S

Figure 3.1 – Exemple de corpus simple et de corpus tabulaire. Le tableau de gauche correspond à un corpus simple et celui de droite à un corpus étiqueté de format tabulaire.

Méta-caractère astérisque (∗) des expressions régulières

Le méta-caractère astérisque (`*`) permet de décrire un nombre quelconque de fois (zéro fois compris) le caractère précédent. Ainsi, l'expression régulière `"e.*e"` décrit toutes les chaînes contenant au moins deux **e**. Donc `"e.*e"` décrit les chaînes *abs**ence**, absurd**eme**nt* de l'ensemble $C1$.

Méta-caractères chapeau (^) et dollar ($) des expressions régulières

Ces deux méta-caractères désignent respectivement le début et la fin d'une chaîne. Par exemple, l'expression régulière `"t$"` décrit toutes les chaînes se finissant par le caractère **t**. Donc `"t$"` décrit les chaînes *absentai**t**, absurdemen**t*** de l'ensemble $C1$.

Méta-caractère anti-slash (\) des expressions régulières

Les méta-caractères ont une signification particulière. Pour y faire référence en tant que simple caractère, il faut les faire précéder du caractère anti-slash (\). Ainsi, l'expression régulière décrivant toutes les chaînes de caractères contenant le caractère astérisque (`*`) s'écrira *"*"*.

Corpus : la source d'information

La bibliothèque LoX travaille sur des corpus écrits. Dans le cas d'un corpus simple, nous n'avons accès qu'aux mots proprement dits du corpus. Dans le cas d'un corpus tabulaire (*i.e.* étiqueté), nous avons accès aux mots ainsi qu'aux étiquettes associées aux mots. Dans ces deux types de corpus, les mots proprement dits (la forme brute des mots dans le texte) sont considérés comme des étiquettes à part entière. Ainsi, un corpus simple est une succession d'étiquettes (une étiquette par mot), et un corpus tabulaire une succession d'ensembles d'étiquettes puisqu'à chacun des mots sont associées plusieurs étiquettes.

La figure 3.1 montre un exemple de corpus simple et un exemple de corpus tabulaire. Le tableau de gauche correspond à un corpus simple. Il n'y a qu'une colonne contenant des mots, cette colonne est baptisée `word`. Le tableau de droite correspond à un corpus étiqueté de format tabulaire. Il comporte trois colonnes. La première, baptisée `jeton` contient la forme brute des mots du texte. La seconde, baptisée `lemme`, contient la forme

lemmatisée des mots. La troisième, baptisée `ems`, contient l'étiquette morphosyntaxique des mots.

Expression logique sur les mots du corpus

Une expression logique est toujours soit vraie soit fausse. Nous allons écrire des expressions logiques qui portent sur les étiquettes des mots du corpus. Supposons que nous voulions écrire une expression logique vraie quand le mot est le verbe *entraîner*. Le mot est le verbe *entraîner* quand l'étiquette *entraîner* se trouve dans la colonne `lemme`. Dans une expression logique, les variables permettent de faire référence aux étiquettes. Par exemple, dans le cas du corpus tabulaire, pour se référer à la colonne contenant les lemmes, il suffit d'écrire `lemme`. `lemme` est la variable contenant la chaîne de la colonne `lemme` et de la ligne du mot que nous cherchons à décrire. Le signe permettant de dire *la chaîne doit être égale à* est tout simplement « `=` ». L'expression logique s'écrit donc :

```
lemme="entraîner".
```

Cette expression est vraie pour tous les mots qui sont une forme du verbe *entraîner*. La puissance expressive des expressions logiques sur les mots commence à se faire entrevoir : l'opération est bien moins fastidieuse que la description de toutes les formes fléchies du verbe *entraîner* avec une simple expression régulière. Sans compter qu'un tel travail ne nous affranchirait pas des ambiguïtés d'homophonie.

Supposons maintenant que nous voulions écrire une expression logique vraie quand le mot est un nom commun. Le mot est un nom commun quand l'étiquette se trouvant dans la colonne `ems` commence par *NC*. Dans une expression logique, le signe permettant de dire *la chaîne doit être décrite par l'expression régulière* est « `~` ». C'est l'équivalent du signe « `=` » pour les expressions régulières. L'expression logique s'écrit donc :

```
ems~"^NC".
```

Dernier exemple : comment écrire une formule logique vraie quand le mot est un verbe différent du verbe *être* et du verbe *avoir* ? La formule logique vraie quand le mot est un verbe est :

```
ems~"^V".
```

La formule logique vraie quand le mot est le verbe *être* ou *avoir* est indifféremment

```
lemme="être" | lemme="avoir"
```

en se plaçant au niveau des formules logiques sur les variables et les chaînes ou

```
lemme~"(^être$|^avoir$)"
```

en utilisant le pouvoir expressif des expressions régulières. Comme nous venons de le voir, dans une formule logique, le *ou* s'écrit « `|` ». Le *et* s'écrit « `&` » et le *non* « `!` ». La formule logique que nous cherchons à écrire est donc indifféremment l'une des formules suivantes :

- `ems~"^V" & !(lemme="être" | lemme="avoir")` ;
- `ems~"^V" & lemme!="être" & lemme!="avoir"` ;
- `ems~"^V" & !(lemme~"(^être$|^avoir$)")` ;
- `ems~"^V" & lemme!~"(^être$|^avoir$)")`.

Méta-expression régulière atomique (MERA)

Une méta-expression régulière atomique (MERA) est une formule logique sur les noms de propriétés, les chaînes de caractères et les expressions régulières. Un mot est décrit par la MERA si l'expression logique associée est vraie pour le mot en question.

Une MERA commence toujours par le délimiteur « `[` » et finit par « `]` ». Ce sont ces délimiteurs qui permettent de dire où commence et où finit la MERA.

Ainsi, pour écrire une MERA qui décrit tous les noms communs du corpus, il suffit d'écrire `[<expression_logique>]` où l'expression logique `<expression_logique>` doit être vraie quand le mot est un nom commun. La MERA s'écrit donc :

```
[ems~"^NC"] .
```

Méta-expression régulière élémentaire (MERE)

Il est possible d'adjoindre un répétiteur pour répéter la MERA un certain nombre de fois. Un répétiteur est de la forme `{`x`,`y`}`, avec $x \leq y$, où x est le nombre minimum de fois où la MERA doit être répétée et y le nombre maximum de fois. Ainsi

```
[ems~"^NC"]{1,3}
```

décrit une suite de un à trois noms communs consécutifs. Si dans le corpus se trouvent deux noms communs consécutifs, l'expression précédente décrit trois portions : la première correspondant au premier nom commun, la seconde correspondant aux deux noms communs consécutifs et la dernière correspondant au second nom commun.

Il existe des répétiteurs prédéfinis :

- `*`, pour dire un nombre quelconque de fois ;
- `+`, pour dire au moins une fois ;
- `?`, pour dire zéro ou une fois.

Il est enfin possible de nommer une MERA pour faire référence ultérieurement au mot décrit par celle-ci. Voici un exemple de MERA nommée :

```
cible:[ems~"^NC"].
```

Dans cet exemple, il est ultérieurement (*i.e.* dans la suite de la requête ou dans un masque) possible de faire référence au nom commun décrit par la MERA en utilisant la variable `cible`.

L'ensemble constitué par la MERA, son nom et son répétiteur associés est appelé meta-expression régulière élémentaire (MERE).

Méta-expression régulière simple (MERS)

La plupart du temps, notre recherche ne se focalise pas sur un mot isolé mais plutôt sur la succession de plusieurs mots. Tandis que répétiteur de MERA permet de s'intéresser à des successions de mots identiques, le formalisme des MERS permet de s'intéresser à des successions de mots distincts.

Par exemple, pour rechercher les noms communs suivis d'un verbe, il est possible d'accoler deux MERE pour constituer une MERS :

```
[ems~"^NC"][ems~"^V"].
```

Cette MERS décrit toutes les successions de deux mots dont le premier est un nom commun et le second un verbe.

Nous pouvons également chercher tous les noms communs suivis d'un adjectif situé dans les quatre mots qui suivent le nom commun. Cette MERS s'écrit :

```
[ems~"^NC"][]{0,3}[ems~"^ADJ"].
```

`[]` est une MERA qui décrit n'importe quel mot. Le multiplicateur `{0,3}` permet de dire que nous acceptons n'importe quel mot de zéro à trois fois consécutives. Nous pouvons imposer que les mots se trouvant entre le nom commun et l'adjectif ne soient pas des verbes. La MERS s'écrit alors :

```
[ems~"^NC"][ems!~"^V"]{0,3}[ems~"^ADJ"].
```

Il existe encore deux formalismes au-dessus des MERS : les *MER* et les *requêtes*. La description de ces formalismes sort du cadre d'une simple présentation. Les MERS sont déjà des requêtes. Les méta-expressions régulières au sens large (MER) élargissent le pouvoir expressif des MERS en ajoutant le parenthèsage, les opérateurs de conjonction (`&`) et de disjonction (`|`) et les multiplicateurs sur les MERS. Les requêtes élargissent encore le pouvoir expressif des MER en ajoutant la clause *stop* et en permettant de préciser sur quelle portion de la MER doit commencer la recherche de correspondance avec le corpus.

Parallèle entre méta-expression régulière simple et expression régulière

Une méta-expression régulière est aux mots ce qu'une expression régulière est aux caractères. Autrement dit, une méta-expression régulière est une expression régulière sur les mots. Une MERA est, dans une méta-expression régulière, ce qu'un caractère est dans une expression régulière.

Prenons, par exemple l'expression régulière suivante :

`la+c`.

Cette dernière décrit toute chaîne de caractères contenant par un **l**, suivi de un à plusieurs **a**, suivie d'un **c**. Prenons maintenant la méta-expression régulière suivante :

`[lemme="le"][ems~"^NC"]+[ems~"^V"]`.

Elle décrit des portions de texte commençant par un mot dont le lemme est *le* suivi de un à plusieurs noms communs suivis d'un verbe. Ces deux expressions peuvent être mises en parallèle comme le montre le tableau 3.1

l	a	+	c
[lemme="le"]	[ems~"^NC"]	+	[ems~"^V"]

Tableau 3.1 – Une expression régulière et une MERS misent en parallèle.

La première expression est une expression régulière sur les caractères qui décrit un ensemble de chaînes de caractères. La deuxième expression est une expression régulière sur les mots qui décrit un ensemble de successions de mots.

Masque

Une requête permet de décrire de manière formelle un phénomène linguistique. Une instance de ce phénomène dans un corpus, c'est-à-dire une portion de corpus décrite par la requête, est appelée une correspondance. À une correspondance sont associées deux positions dans le corpus (début et fin de la correspondance), ainsi qu'un certain nombre de références à des mots de la correspondance. Les masques permettent à l'utilisateur de préciser comment la correspondance doit être utilisée pour générer une chaîne de caractères.

Conventions sur la définition formelle de la syntaxe des formalismes

Les sections qui suivent (3.4.4 à 3.4.12) donnent une définition détaillée et formelle de chacun des niveaux du formalisme de la bibliothèque LoX. Les définitions formelles de la syntaxe des formalismes sont données sous formes de règles de réécriture :

1. `<MER>` ← `<MERS>`
2. `<MER>` ← **(**`<MER>`**)**

Elles sont suivies d'une explication détaillée comme ci-dessous.

1. La première ligne `<MER>` ← `<MERS>` doit se lire : `<MERS>` est une (ou peut se réécrire en) `<MER>`.
2. La deuxième ligne doit se lire : `<MER>` entre parenthèses est une (ou peut se réécrire en) `<MER>`.

Dans les règles de réécriture, `<formalisme>` désigne le formalisme `formalisme`. En revanche, les caractères en gras font réellement partie du formalisme. Ainsi, un exemple de **`(`**`<MER>`**`)`** est **`(`**`[ems~"^NC"]`**`)`** où `<MER>` désigne la MER `[ems~"^NC"]`.

Remarque

Dans la bibliothèque LoX, les mots prédéfinis (comme *word*, *begin*, *end*, etc.) sont en anglais. Ce choix n'a pas été dicté pour des raisons d'universalité, de suprématie de la langue anglaise ou d'insurrection contre la loi Toubon. Il a, au contraire, été retenu pour un usage destiné aux non anglophones de manière à bien différencier les mots du langage (*i.e.* des formalismes) des mots de l'utilisateur (désignation des colonnes, références, etc.).

3.4.4 Chaîne de caractères `<chaine>`

Une chaîne de caractères n'est, ni plus ni moins, qu'une suite de caractères.

Définition 3.3 – Chaîne de caractères `<chaine>` –
Une chaîne de caractères est une suite de caractères.

1. *`<chaine>` ← `<caractere>`*
2. *`<chaine>` ← `<chaine><caractere>`*

Explication de la syntaxe des chaînes :

1. Un caractère est une chaîne de caractères.
2. La concaténation d'une chaîne et d'un caractère est une chaîne.

Comme nous le verrons plus loin, une chaîne de caractères est, dans la plupart des cas, encadrée de guillemets : `"<chaine>"`. Dans ce cas, et seulement dans ce cas, le caractère `\` est un méta-caractère. Il permet de faire référence au caractère qui le suit. Ainsi, la suite de caractères `abc"d` doit s'écrire `abc\"d` quand elle est entre `"` (`"abc\"d"`). Ce caractère est également indispensable pour faire référence aux caractères `[`, `]`, `(`, `)`, et `\`.

3.4.5 Référence `<reference>`, propriété `<propriete>` et variable `<variable>`

Référence `<reference>`

Les références sont toujours définies au sein d'une correspondance. Ce sont des *pointeurs* vers des mots du corpus. À chaque référence est donc associée une position dans le corpus.

Définition 3.4 – Référence `<reference>` –
Une référence pointe sur un mot du corpus.

1. *`<reference>` ←* ***`begin`***
2. *`<reference>` ←* ***`end`***
3. *`<reference>` ←* ***`next`***

4. <reference> ← **prev**
5. <reference> ← **word(**<exp_arith>**)**
6. <reference> ← <chaine>

Explication de la syntaxe des références :

1. `begin` est une référence prédéfinie pointant sur le premier mot de la correspondance.
2. `end` est une référence prédéfinie pointant sur le dernier mot de la correspondance.
3. `next` est une référence prédéfinie pointant sur le mot suivant le mot courant.
4. `prev` est une référence prédéfinie pointant sur le mot précédent le mot courant.
5. `word(<exp_arith>)` est une référence dynamique pointant sur le mot se trouvant à la position `<exp_arith>` dans le corpus (sachant que la position du premier mot du corpus est zéro).
6. `<chaine>` est une référence définie par l'utilisateur. Cette définition se fait en tête d'une MERA (cf. section 3.4.9). Son nom est `<chaine>` où `<chaine>` est une chaîne de caractères alphanumériques autre que les chaînes prédéfinies `begin`, `end`, `prev`, `next`, `word`, `index` et `exist`. Pour éviter toute confusion, il est préférable qu'une référence ne porte pas le même nom qu'une propriété.

Propriété <propriete>

Une propriété est une information sur un mot du corpus. Par exemple, le lemme d'un mot constitue une propriété de ce mot. Certaines propriétés sont prédéfinies et toujours disponibles. D'autres sont fonction du corpus. Par exemple, lorsque le corpus utilisé est un corpus annoté au format tabulaire, l'utilisateur nomme chacune des colonnes de ce corpus. Chaque colonne correspond alors à une propriété.

Définition 3.5 – Propriété <propriete> –
Une propriété est une information sur un mot du corpus.

1. <propriete> ← **index**
2. <propriete> ← **exist**
3. <propriete> ← **word**
4. <propriete> ← <chaine>

Explication de la syntaxe des propriétés :

1. `index` est un nom de propriété réservé permettant d'accéder à la position absolue d'un mot dans le corpus (sachant que la position du premier mot du corpus est zéro).
2. `exist` est un nom de propriété réservé permettant de savoir si une référence désigne bien un mot ou non.
3. `word` est un nom de propriété réservé permettant d'accéder à la chaîne correspondant à la forme brute d'un mot dans le cadre de l'utilisation d'un corpus de type texte brut.
4. `<chaine>` est le nom d'une propriété des mots définie par l'utilisateur, par exemple, un nom de colonne dans le cas des corpus tabulaires.

Variable <variable>

Une variable est la propriété particulière d'une référence ou du mot courant. Par exemple, si le corpus de travail comporte une propriété lemme, la chaîne de caractères correspondant au lemme du premier mot d'une correspondance est accessible par la variable : `begin.lemme`.

Définition 3.6 – Variable <variable> –
Une variable est une information sur un mot du corpus.

1. *<variable>* ← *<reference>.<propriete>*
2. *<variable>* ← *<propriete>*

Explication de la syntaxe des variables :

1. `<reference>.<propriete>` permet d'accéder à la propriété `<propriete>` de la référence `<reference>`.
2. `<propriete>`, sans préciser un nom de référence, permet d'accéder à une propriété du mot courant. Ceci est possible chaque fois qu'une position par défaut est disponible. C'est le cas dans le corps d'une MERA où la position par défaut est celle du mot que la MERA cherche à décrire (le mot courant). C'est aussi le cas dans la partie `<masque>` du masque de forme `[B:<masque>;<exp_arith>;<exp_arith>]` où la position par défaut correspond à la valeur du compteur (cf. section 3.4.12).

Type des variables

Une variable de la forme `<chaine>.exist` est forcément du type booléen (expression logique). Toutes les autres variables sont traitées comme des nombres dans les expressions arithmétiques (avec une évaluation dans le cas où la variable désigne une chaîne) et comme des chaînes dans les comparaisons avec des chaînes ou des expressions régulières (avec une conversion dans le cas où la variable désigne un nombre).

3.4.6 Expression arithmétique <exp_arith>

Une expression arithmétique désigne un nombre de manière explicite, ou par l'intermédiaire d'une formule.

Définition 3.7 – Expression arithmétique <exp_arith> –
Une expression arithmétique désigne un nombre.

1. *<exp_arith>* ← *<nombre>*
2. *<exp_arith>* ← *<variable>*
3. *<exp_arith>* ← **(***<exp_arith>***)**
4. *<exp_arith>* ← **-***<exp_arith>*
5. *<exp_arith>* ← *<exp_arith>***+***<exp_arith>*
6. *<exp_arith>* ← *<exp_arith>***-***<exp_arith>*
7. *<exp_arith>* ← *<exp_arith>******<exp_arith>*
8. *<exp_arith>* ← *<exp_arith>*/*<exp_arith>*
9. *<exp_arith>* ← *<exp_arith>*^*<exp_arith>*
10. *<exp_arith>* ← ***frac(****<exp_arith>***)**
11. *<exp_arith>* ← ***ent(****<exp_arith>***)**
12. *<exp_arith>* ← ***rnd(****<exp_arith>***)**

13. <exp_arith> ← **exp(**<exp_arith>**)**
14. <exp_arith> ← **ln(**<exp_arith>**)**
15. <exp_arith> ← **log(**<exp_arith>**)**
16. <exp_arith> ← **abs(**<exp_arith>**)**
17. <exp_arith> ← **sqr(**<exp_arith>**)**
18. <exp_arith> ← **random(**<exp_arith>**)**
19. <exp_arith> ← **pi**
20. <exp_arith> ← **or**
21. <exp_arith> ← **e**

Explication de la syntaxe des expressions arithmétiques :

1. Un nombre est une expression arithmétique.
2. Correspond à la valeur numérique de la variable.
3. Règle des parenthèses : correspond au nombre retourné par `<exp_arith>`.
4. Opérateur unaire retournant l'opposé de `<exp_arith>`.
5. Addition de deux expressions arithmétiques.
6. Soustraction de deux expressions arithmétiques.
7. Multiplication de deux expressions arithmétiques.
8. Division de deux expressions arithmétiques.
9. Mise à la puissance.
10. Partie décimale de l'expression `<exp_arith>`.
11. Partie entière de l'expression `<exp_arith>`.
12. Arrondi de l'expression `<exp_arith>`.
13. Exponentielle de l'expression `<exp_arith>`.
14. Logarithme népérien de l'expression `<exp_arith>`.
15. Logarithme décimal de l'expression `<exp_arith>`.
16. Valeur absolue de l'expression `<exp_arith>`.
17. Racine carrée de l'expression `<exp_arith>`.
18. Nombre aléatoire entre zéro et un avec comme graine `<exp_arith>`.
19. Constante pi.
20. Nombre d'or.
21. Constante e ($e = e^1 =$`exp(1)`).

3.4.7 Expression régulière `<exp_reg>`

Le formalisme des expressions régulières permet de décrire des ensembles de chaînes de caractères. Il s'agit ici des expressions régulières supportées par la commande *egrep* sous *Unix/Linux*.

Définition 3.8 – Expression régulière `<exp_reg>` –
Une expression régulière décrit des ensembles de chaînes de caractères.

1. <exp_reg> ← <caractere>
2. <exp_reg> ← **.**
3. <exp_reg> ← ****<caractere>
4. <exp_reg> ← **[**<liste_de_caracteres>**]**
5. <exp_reg> ← <exp_reg>*****

6. <exp_reg> ← <exp_reg>+
7. <exp_reg> ← (<exp_reg>)
8. <exp_reg> ← <exp_reg><exp_reg>
9. <exp_reg> ← (<exp_reg>|<exp_reg>)
10. <exp_reg> ← ^<exp_reg>
    <exp_reg> ← <exp_reg>$

Explication de la syntaxe des expressions régulières :

1. Un caractère est une expression régulière qui se désigne lui-même, excepté pour les caractères `.`, `?`, `+`, `{`, `|`, `(`, `)`, `^`, `$`, `\`, `[` et `]` qui sont des méta-caractères et ont une signification spéciale ; pour désigner ces méta-caractères, il faut les faire précéder d'un antislash (\).
2. **`.`** est un méta-caractère qui désigne n'importe quel caractère.
3. **`\`** décrit le caractère qui suit, exception faite des caractères 1, 2, . . . , 9, **`<`** et **`>`**. Le caractère **`\`** permet ainsi de décrire des méta-caractères.
4. **`[`**`<liste_de_caracteres>`**`]`** décrit l'un des caractères de la liste de caractères, par exemple **`[abcdf]`** décrit les caractères **a**, **b**, **c**, **d** ou **f**. Le caractère **`-`** permet de décrire des ensembles de caractères consécutifs, par exemple **`[a-df]`** est équivalent à **`[abcdf]`**. La plupart des méta-caractères perdent leur signification spéciale dans une liste. Pour insérer un `]` dans une liste, il faut le mettre en tête de liste, pour inclure un `^`, il faut le mettre n'importe où sauf en tête de liste, `-` se place à la fin de la liste. Lorsque **`[`**`<liste_de_caracteres>`**`]`** est de la forme **`[^`**`<liste_de_caracteres>`**`]`**, cette expression ne décrit plus l'un des caractères de la liste de caractères `<liste_de_caracteres>`, mais tous les caractères qui ne sont pas dans la liste de caractères `<liste_de_caracteres>`.
5. Décrit aucune ou plusieurs occurrences de l'expression régulière qui précède le caractère **`*`**. Attention, dans l'expression `<exp_reg>`**`*`**, l'expression régulière qui précède le caractère **`*`** n'est pas `<exp_reg>` mais la plus petite sous-expression régulière (qui n'est pas de la forme `<exp_reg><exp_reg>`) la plus à droite de `<exp_reg>`.
6. `<exp_reg>`**`+`** est équivalent à `<exp_reg>`**`*`**, excepté que l'expression régulière doit être décrite au moins une fois (donc une ou plusieurs fois).
7. Une expression régulière entre parenthèses **`(`**`<exp_reg>`**`)`** décrit ce que décrit `<exp_reg>`.
8. Décrit une chaîne constituée de la concaténation de deux sous-chaînes décrites par chacune des expressions régulières `<exp_reg>`.
9. Décrit toute chaîne décrite par la première expression régulière ou la deuxième.
10. **`^`** en tête d'une expression régulière désigne le début de la chaîne, **`$`** en queue d'une expression régulière désigne la fin de la chaîne.

3.4.8 Expression logique `<exp_log>` et Méta-expression régulière atomique `<MERA>`

Expression logique

Définition 3.9 – Expression logique `<exp_log>` –
Une expression logique est soit vraie soit fausse, et porte sur les étiquettes des mots du corpus.

1. <exp_log> ← ***true***

2. <exp_log> ← **false**
3. <exp_log> ← **(**<exp_log>**)**
4. <exp_log> ← <exp_arith>**=**<exp_arith>
5. <exp_log> ← <exp_arith>**!=**<exp_arith>
6. <exp_log> ← <exp_arith>**<**<exp_arith>
7. <exp_log> ← <exp_arith>**>**<exp_arith>
8. <exp_log> ← <exp_arith>**<=**<exp_arith>
9. <exp_log> ← <exp_arith>**>=**<exp_arith>
10. <exp_log> ← <exp_log>**&**<exp_log>
11. <exp_log> ← <exp_log>**|**<exp_log>
12. <exp_log> ← **!**<exp_log>
13. <exp_log> ← <variable>
14. <exp_log> ← **"**<chaine>**"="**<chaine>**"**
    <exp_log> ← **"**<chaine>**"!="**<chaine>**"**
    <exp_log> ← <variable>**="**<chaine>**"**
    <exp_log> ← <variable>**!="**<chaine>**"**
    <exp_log> ← **"**<chaine>**"=**<variable>
    <exp_log> ← **"**<chaine>**"!=**<variable>
    <exp_log> ← <variable>**="**<variable>**"**
    <exp_log> ← <variable>**!="**<variable>**"**
15. <exp_log> ← **"**<chaine>**"~"**<exp_reg>**"**
    <exp_log> ← **"**<chaine>**"!~"**<exp_reg>**"**
    <exp_log> ← <variable>**~"**<exp_reg>**"**
    <exp_log> ← <variable>**!~"**<exp_reg>**"**
    <exp_log> ← <variable>**~**<variable>
    <exp_log> ← <variable>**!~**<variable>

Explication de la syntaxe des expressions logiques :

1. Tautologie (toujours vraie).
2. Antitautologie (toujours fausse).
3. Règle des parenthèses.
4. Test d'égalité entre deux expressions arithmétiques `<exp_arith>`. Si les deux expressions arithmétiques sont des variables (`<variables>`), elles sont comparées en terme de chaînes de caractères et non en terme de nombres, sans que cela ne pose de problème.
5. Test de non-égalité entre deux expressions arithmétiques `<exp_arith>`.
6. Test d'infériorité stricte entre expressions arithmétiques `<exp_arith>`.
7. Test de supériorité stricte entre expressions arithmétiques `<exp_arith>`.
8. Test d'infériorité entre expressions arithmétiques `<exp_arith>`.
9. Test de supériorité entre expressions arithmétiques `<exp_arith>`.
10. Conjonction logique entre deux expressions logiques `<exp_log>`.
11. Disjonction logique entre deux expressions logiques `<exp_log>`.
12. Négation logique de l'expression logique `<exp_log>`.
13. Équivalant à la valeur logique de la variable `<variable>`.
14. Teste l'égalité (`=`) ou l'inégalité (`!=`) entre deux chaînes. Une chaîne peut être littérale (`"<chaine>"`) ou désignée par une variable (`<variable>`). Dans le cas de l'égalité ou de l'inégalité entre deux variables, une ambiguïté subsiste : les

variables désignent-elle des chaînes, ou des nombres ? Le problème est résolu en assimilant toujours les variables à des chaînes. Ainsi, s'il s'agit de nombres, ils sont convertis en chaîne et le test reste correct.

15. Teste si l'expression régulière (à droite de l'opérateur) décrit (`~`), ou ne décrit pas (`!~`) la chaîne (à gauche de l'opérateur). Une chaîne peut être littérale (`"<chaine>"`) ou désignée par une variable (`<variable>`). Une expression régulière peut être littérale (`"<exp_reg>"`) ou implicite (`<variable>`). Ce qu'il faut retenir c'est que l'expression qui est assimilée à la chaîne est à gauche et l'expression qui est assimilée à une expression régulière est à droite.

Méta-expression régulière atomique `<MERA>`

Définition 3.10 – Méta-expression régulière atomique `<MERA>` –
*Une méta-expression régulière atomique (*MERA*) est une expression logique entre crochets `[]`. Un mot est décrit par la* MERA *si l'expression logique associée est* vraie *pour le mot en question.*

- `<MERA>` ← `[<exp_log>]`

Remarque : `[]` est une MERA spéciale décrivant n'importe quel mot. En fait `[]` est équivalent à `[true]`.

3.4.9 Méta-expression régulière élémentaire `<MERE>` et méta-expression régulière simple `<MERS>`

Méta-expression régulière élémentaire

Définition 3.11 – Méta-expression régulière élémentaire `<MERE>` –
Une MERE *se décompose en trois parties. La première partie est facultative et permet de générer une référence (<**chaine**>). La seconde, constituant le corps de la* MERE*, est une* MERA*. La dernière, facultative, constitue le multiplicateur `{<exp_arith>,<exp_arith>}`.*

- `<MERE>` ← `<MERA>`
- `<MERE>` ← `<MERA>{<exp_arith>,<exp_arith>}`
- `<MERE>` ← `<chaine>:<MERA>`
- `<MERE>` ← `<chaine>:<MERA>{<exp_arith>,<exp_arith>}`

Référence

Il s'agit d'une chaîne qui permet de faire ultérieurement référence au ou aux mots que décrit la MERE (cf. la section 3.4.5, notamment pour connaître les restrictions sur l'écriture de `<chaine>`).

Si la MERE n'est pas sous la portée d'un multiplicateur, le nom de la référence est tout simplement celui donné par l'utilisateur, soit `<chaine>`. En revanche, il se peut que la MERE décrive plusieurs mots, soit consécutifs (généralement à cause du multiplicateur de la MERE), soit non consécutifs (possible si la MERE se trouve sous la portée de un ou plusieurs multiplicateurs de MER, cf. section 3.4.10). Dans le cas où la MERE se trouve sous la portée de un ou plusieurs multiplicateurs, une référence distincte est générée pour chacun des mots décrits. Il faut donc altérer le nom donné par l'utilisateur pour générer autant de noms (donc de références) qu'il y a de mots à référencer. Cette altération se fait par l'adjonction de un ou plusieurs suffixes. Si le corps de la MERE est sous la portée de n multiplicateurs

il y aura n suffixes. Le suffixe le plus proche de la chaîne `<chaine>` correspond au multiplicateur le plus proche. Chaque suffixe est de la forme `_<indice>`. `<indice>` est un entier représentant la valeur courante, pour le mot pointé, du compteur de multiplicateur.

Répétiteur classique

Il est possible d'adjoindre un répétiteur pour répéter la MERA un certain nombre de fois. Un répétiteur classique est de la forme `{`x`,`y`}`, avec $x \leq y$, où x est le nombre minimum de fois où la MERA doit être répétée et y le nombre maximum de fois.

Répétiteurs prédéfinis

Trois répétiteurs sont prédéfinis :

`*` , pour dire un nombre quelconque de fois, est équivalent à `{0, LONG_MAX}` ;

`+` , pour dire au moins une fois, est équivalent à `{1, LONG_MAX}` ;

`?` , pour dire zéro ou une fois, est équivalent à `{0, 1}`.

`LONG_MAX` est égal au plus grand entier standard qu'il est possible de représenter en `C++`.

Répétiteur préférentiel

Le répétiteur préférentiel est un répétiteur particulier, sa forme est `{`x`,`y`}` avec $x > y$. Ce répétiteur peut être lu comme : la MERA doit être répétée x fois si possible, sinon $x-1$ fois, sinon $x-2$ fois, ... mais au minimum y fois.

Par exemple, supposons la séquence de mots $x \overset{1}{a}\overset{2}{a}\overset{3}{a}\ y$ et un sens d'évaluation de gauche à droite (nous verrons dans la section section 3.4.10 qu'il est possible d'agir sur le sens de l'évaluation). Sur cette séquence, la requête $[a]\{1,4\}$ retourne six correspondances ($\overset{1}{a}$, $\overset{12}{aa}$, $\overset{123}{aaa}$, $\overset{2}{a}$, $\overset{23}{aa}$, $\overset{3}{a}$), tandis que la requête $[a]\{4,1\}$ en retourne trois ($\overset{123}{aaa}$, $\overset{23}{aa}$, $\overset{3}{a}$).

Remarque

Pour les répétiteurs classiques et préférentiels, x et y sont des expressions arithmétiques évaluées une fois pour toute. Ces expressions ne peuvent donc pas contenir de variables.

Méta-expression régulière simple `<MERS>`

Définition 3.12 – Méta-expression régulière simple `<MERS>` –
*Une méta-expression régulière simple (*MERS*) permet de concaténer des* MERE *pour décrire des successions de mots distincts.*

1. *`<MERS>` ← `<MERE>`*
2. *`<MERS>` ← `<MERS><MERS>`*

Explication de la syntaxe des méta-expressions régulières simples :

1. Une MERE est une MERS.
2. La juxtaposition de deux MERS est également une MERS.

Une MERS est donc la juxtaposition d'un nombre quelconque de MERE.

3.4.10 Méta-expression régulière `<MER>`

Définition 3.13 – Méta-expression régulière `<MER>` –
*Les méta-expressions régulières (*MER*) élargissent le pouvoir expressif des* MERS *en ajoutant le parenthèsage, les opérateurs de conjonction (`&`) et de disjonction (`|`) et*

les multiplicateurs sur les MERS.

1. `<MER>` ← `<MERS>`
2. `<MER>` ← **`(`**`<MER>`**`)`**
3. `<MER>` ← **`(`**`<MER>`**`){`**`<exp_arith>,<exp_arith>`**`}`**
   `<MER>` ← **`(`**`<MER>`**`)*`**
   `<MER>` ← **`(`**`<MER>`**`)+`**
   `<MER>` ← **`(`**`<MER>`**`)?`**
4. `<MER>` ← `<MER><MER>`
5. `<MER>` ← `<MER>` **`|`** `<MER>`
6. `<MER>` ← `<MER>` **`&`** `<MER>`
7. `<MER>` ← `<MER>` **`!&`** `<MER>`

Explication de la syntaxe des méta-expressions régulières :

1. Une MERS est une MER.
2. Règle des parenthèses.
3. Les MER entre parenthèses supportent les multiplicateurs. La syntaxe et la signification de ces multiplicateurs est équivalente à celle des MERE présentées section 3.4.9.
4. La concaténation de deux MER est une MER.
5. La disjonction de deux MER est une MER. Concrètement, lors d'une évaluation, il s'agit de l'union des correspondances des deux MER. Si n_1 est le nombre de correspondances de la première MER et n_2 celui de la deuxième, le résultat produira $(n_1 + n_2)$ correspondances.
6. La conjonction de deux MER est une MER. Pour faire simple cette conjonction produit ce qu'elle doit produire (c'est la façon la plus simple de le dire, une autre façon est exposée ci-dessous).

   Soit c_1 l'ensemble des correspondances générées par la MER de gauche et c_2 l'ensemble des correspondances générées par la MER de droite. Soit c'_1 l'ensemble des correspondances de c_1 qui ont une correspondance similaire dans c_2 et c'_2 l'ensemble des correspondances de c_2 qui ont une correspondance similaire dans c_1. Soit S_1 ($S_1 = \{s_{11}, s_{12}, \ldots, s_{1n}\}$) l'ensemble des sous-ensembles de correspondances similaires de c'_1 et S_2 ($S_2 = \{s_{21}, s_{22}, \ldots, s_{2n}\}$) l'ensemble des sous-ensembles de correspondances similaires de c'_2 classées dans le même ordre que S_1. En définissant le produit de deux correspondances similaires comme une nouvelle correspondance similaire comportant toutes les références des deux correspondances initiales et en notant $\times$ le produit entre deux ensembles de correspondances similaires contenant toutes deux des correspondances similaires, le résultat c devient : $c = \{s_{11} \times s_{21},\ s_{12} \times s_{22}, \ldots,\ s_{1n} \times s_{2n}\}$.
7. Le résultat est l'ensemble des correspondances de la MER de gauche qui n'ont pas de correspondance similaire dans la MER de droite.

3.4.11 Requête `<requete>`

Définition 3.14 – Requête `<requete>` –
Les requêtes enrichissent le formalisme des MER *d'une clause* stop *permettant d'interrompre la recherche de correspondance et permettent de préciser sur quelle portion de la* MER *doit commencer la recherche de correspondance avec le corpus.*

1. `<requete>` ← `<MER>`

2. *<requete>* ← *<MER>* ***stop (****<exp_log>****)***
3. *<requete>* ← ***<<****MER****>>****<MER>*
   <requete> ← ***<<****MER****>>****<MER>* ***stop (****<exp_log>****)***
   <requete> ← *<MER>****<<****MER****>>***
   <requete> ← *<MER>****<<****MER****>>*** ***stop (****<exp_log>****)***
   <requete> ← *<MER>****<<****MER****>>****<MER>*
   <requete> ← *<MER>****<<****MER****>>****<MER>* ***stop (****<exps_log>****)***

Explication de la syntaxe des requêtes :

1. Une MER est une requête.
2. Une requête peut toujours se terminer par la spécification d'une clause *stop*. Cette clause permet de préciser une condition qui interrompt la recherche et rejette la tentative de correspondance en cours. L'expression logique de la clause *stop* ne peut pas faire référence à d'autres variables que celle de la forme :
   - `<propriete>`,
   - `begin.<propriete>`,
   - `end.<propriete>`,
   - `next.<propriete>`,
   - `prev.<propriete>`.
3. Certaines requêtes possèdent une MER entre `<>` (`<<MER>>`). Nous appelons MER centrale une telle MER. La tentative de correspondance entre la ou les MER de la requête et une portion de corpus se fait toujours en partant d'une position donnée dans le corpus et en progressant de gauche à droite dans le corpus et la ou les MER de la requête. Les `<>` permettent de moduler cette procédure. Dans une requête, s'il existe une MER centrale, entre `<>`, c'est la correspondance entre le corpus et la MER centrale qui est recherchée en premier. Ensuite, s'il existe une MER à gauche de la MER centrale nous recherchons sa correspondance mais de droite à gauche cette fois. Enfin, s'il existe une MER à droite de la MER centrale, nous recherchons sa correspondance de manière classique. Cette façon de procéder possède deux justifications : la première est que c'est la seule façon d'écrire certaine requête (pouvoir expressif) ; la seconde est que c'est souvent un moyen d'écrire des requêtes qui seront évaluées bien plus rapidement (technique d'optimisation). La portion entre `<>` ne doit pas être sous la portée de parenthèses, ce qui n'implique pas qu'elle ne puisse en contenir.

Exemple d'évaluation et d'optimisation d'une requête

La requête suivante décrit des n-grammes (cf. définition 7.2 page 158), avec $n = 5$, contenant le lemme *chef*, ne contenant pas de ponctuation et commençant et finissant par un nom commun, un adverbe, un verbe ou un adjectif.

```
([ems~"(^NC|^ADV|^V|^ADJ)"][ems!~"^PCT"]{0,3})?
[lemme="chef"]
([ems!~"^PCT"]{0,3}[ems~"(^NC|^ADV|^V|^ADJ)"])?
stop((end.index-begin.index)>4)
```

Figure 3.2 – Exemple de requête décrivant des 5-grammes.

Cette requête est analysée de gauche à droite. Le corpus est également parcouru de gauche à droite. Ainsi, la requête va tout d'abord tenter de décrire une portion du

corpus commençant par le premier mot du corpus, puis par le second, etc. La première MERE (`[ems~"(^NC|^ADV|^V|^ADJ)"]`) de la requête est souvent vérifiée car de nombreux mots sont un nom commun, un adverbe, un verbe ou un adjectif. La seconde MERE (`[ems!~"^PCT"]{0,3}`) décrit généralement quatre portions de corpus si les trois mots suivant le mot décrit par la première MERE ne sont pas des ponctuations (le cas est également très fréquent). Ce n'est que sur la troisième MERE de la requête (`[lemme="chef"]`) que l'application de la requête échoue dans la majorité des cas (un mot ayant pour lemme chef est un phénomène rare dans un corpus). Ainsi, appliquée sur un corpus de taille importante, l'évaluation de cette requête est extrêmement lente. Il serait plus pertinent de rechercher les mots dont le lemme est *chef* puis de continuer l'application de la requête autour de ces mots. Pour cela, il suffit d'encadrer par les délimiteurs `<` et `>` la partie de la requête devant être évaluée en premier. La requête devient donc :

```
([ems~"(^NC|^ADV|^V|^ADJ)"][ems!~"^PCT"]{0,3})?
<[lemme="chef"]>
([ems!~"^PCT"]{0,3}[type_gram~"(^NC|^ADV|^V|^ADJ)"])?
stop((end.index-begin.index)>4)
```

Figure 3.3 – Même requête que celle de la figure 3.2 mais optimisée à l'aide des délimiteurs `<` et `>`.

Dans une telle requête, la partie entre `<>` est évaluée en premier. Si l'évaluation réussit, la partie précédente est évaluée à rebours (de droite à gauche) tandis que la partie suivante est évaluée de manière classique (de gauche à droite).

Attention, le sens de l'évaluation est important dans le cas de l'utilisation du répétiteur préférentiel.

Remarque sur l'utilisation du répétiteur préférentiel

Les deux requêtes que nous venons de voir ne s'évaluent pas dans le même intervalle de temps sur un corpus (la seconde est bien plus rapide) mais elles retournent cependant toujours le même résultat. Dans la majorité des cas, la présence ou l'absence des délimiteurs `<>` n'a pas d'impact sur le résultat. Par exemple, les requêtes

```
[word="b"]{1,4} [word="a"] [word="b"]{1,4}
```

et

```
[word="b"]{1,4} <[word="a"]> [word="b"]{1,4}
```

peuvent toutes les deux retourner les six correspondances *bbbab*, *bbbabb*, *bbab*, *bbabb*, *bab*, *babb*.

Il n'en va pas toujours de même, notamment lorsque des délimiteurs `<` et `>` sont introduits dans une expression contenant un répétiteur préférentiel `{`x`,`y`}` avec $x > y$. Soit, par exemple, la séquence de mots $x\ b\ b\ b\ a\ b\ b\ y$. Sur cette séquence, la requête

```
[word="b"]{4,1} [word="a"] [word="b"]{4,1}
```

retourne trois correspondances *bbbabb*, *bbabb*, *babb* tandis que la requête

```
[word="b"]{4,1} <[word="a"]> [word="b"]{4,1}
```

n'en retourne qu'une, *bbbabb*, en raison de l'évaluation à rebours de la partie à gauche du délimiteur `<`. Le répétiteur préférentiel et les délimiteurs `<` et `>` interagissent donc lors de l'évaluation d'une requête.

La clause *stop* peut également produire des effets d'interaction similaires. L'utilisation des délimiteurs `<` et `>` peut modifier le résultat d'une requête, il faut donc les utiliser à bon escient, c'est-à-dire quand :

- ils permettent une optimisation significative de la requête ;
- ils sont nécessaires pour obtenir le résultat escompté.

3.4.12 Masque `<masque>`

Un masque est une expression qui, appliquée sur une correspondance, permet de générer une chaîne de caractères. Pour générer cette chaîne de caractères, nous pouvons spécifier explicitement les caractères à générer, mais nous pouvons également faire référence à certains mots de la correspondance par l'intermédiaire des références.

Définition 3.15 – Masque `<masque>` –
Un masque permet de préciser comment la correspondance doit être utilisée pour générer une chaîne de caractères.

1. `<masque>` ← `<chaine>`
2. `<masque>` ← **`[C:`**`<exp_arith>`**`]`**
3. `<masque>` ← **`[N:`**`<exp_arith>`**`]`**
4. `<masque>` ← **`[P:`**`<variable>`**`]`**
5. `<masque>` ← **`[?:`**`<exp_log>;<masque>;<masque>`**`]`**
6. `<masque>` ← **`[B:`**`<masque>;<exp_arith>;<exp_arith>`**`]`**
   `<masque>` ← **`[B(`**`<chaine>`**`):`**`<masque>;<exp_arith>;<exp_arith>`**`]`**
7. `<masque>` ← **`[W:`**`<masque>;<exp_log>`**`]`**
   `<masque>` ← **`[W(`**`<chaine>`**`):`**`<masque>;<exp_log>`**`]`**
8. `<masque>` ← `<masque><masque>`

Explication de la syntaxe des masques :

1. Une chaîne est un masque.
2. Permet de désigner un caractère dont le code est le résultat de la formule `<exp_arith>`.
3. Écrit le nombre correspondant au résultat de la formule `<exp_arith>`.
4. Écrit la chaîne de caractères désignée par `<variable>`.
5. Équivalent au masque de gauche si `<exp_log>` est vraie et au masque de droite dans le cas contraire.
6. Cette séquence génère une référence par défaut pointant successivement sur toutes les positions allant de la position spécifiée par la première `<exp_arith>` jusqu'à la position spécifiée par la deuxième `<exp_arith>`, en générant à chaque pas la chaîne `<masque>` ; dans la deuxième forme, `<chaine>` est le nom d'une variable dont la valeur varie au cours des itérations de la valeur de la première `<exp_arith>` à celle de la deuxième.
7. Cette séquence permet d'itérer la génération de la chaîne `<masque>` tant que `<exp_log>` est vraie ; dans la deuxième forme, `<chaine>` est le nom d'une variable dont la valeur est un pour la première itération et est incrémentée à chaque itération.
8. La concaténation de deux masques est un masque.

Remarque

Les chaînes suivantes `[C:`, `[N:`, `[P:`, `[?:`, `[B:`, `[B(`, `[W:`, `[W(` sont des méta-chaînes et ne peuvent être utilisées comme de simples chaînes dans un masque. Les espaces qui encadrent un masque sont ignorés. Pour introduire des espaces en fin ou en début, il faut les spécifier explicitement par la syntaxe `[C:<exp_arith>]` soit `[C:32]`, 32 étant le code ASCII du caractère espace.

3.4.13 Description sommaire de l'implémentation

Indépendance vis-à-vis du format des corpus

Notre bibliothèque doit pouvoir s'adapter à nos besoins susceptibles d'évoluer, et doit également être largement réutilisable pour une grande variété d'applications, voire par une grande variété de programmeurs (au sein de notre équipe ou d'autres équipes). Pour ces multiples raisons, cette bibliothèque doit être relativement indépendante du format des corpus sur lesquels elle travaille. Nous avons donc choisi de mettre en œuvre un mécanisme d'abstraction de l'objet corpus.

En `C++`, ce mécanisme d'abstraction se fait par l'intermédiaire des *classes abstraites*. De telles classes ne sont pas destinées à être elles-même directement instanciées, mais seulement à servir de classes de base à d'autres classes (qui, elles, seront instanciées). Nous avons donc créé une classe abstraite *corpus*. Un corpus étant constitué de mots, nous avons également créé une classe abstraite *mot*. Ainsi, chaque programmeur désirant utiliser cette bibliothèque pour créer une application particulière est libre de développer ses propres classes *corpus* et *mot* pour accéder aux informations contenues dans le corpus que l'application est destinée à utiliser. Bien entendu, ces classes *corpus* et *mot* doivent être dérivées des classes abstraites *corpus* et *mot* fournies par la bibliothèque. Grâce à un mécanisme nommé polymorphisme, la bibliothèque utilise les classes développées par le programmeur pour accéder au corpus. Pour nos besoins, nous avons développé deux ensembles de classes dérivées, un pour accéder à des corpus de texte brut et un pour accéder à des corpus tabulaires comme ceux générés par le logiciel Cordial Analyseur.

Une cascade d'outils disponibles

Notre bibliothèque ne rend pas uniquement disponibles les objets terminaux du formalisme mais également tous les formalismes intermédiaires :

- un objet pour les *expressions arithmétiques* (formalisme décrit section 3.4.6) ;
- un objet pour les *expressions logiques* (formalisme décrit section 3.4.8) ;
- un objet pour les *méta-expressions régulières simples* (formalisme décrit section 3.4.9) ;
- un objet pour les *méta-expressions régulières* (formalisme décrit section 3.4.10) ;
- un objet pour les *requêtes* (formalisme décrit section 3.4.11) ;
- un objet pour les *masques* (formalisme décrit section 3.4.12).

Pour réaliser cette bibliothèque, nous avons dû implémenter chacun des objets ci-dessus. Pour chaque objet, nous avons réalisé un analyseur syntaxique (*parser* en anglais) permettant de décomposer la chaîne de caractères correspondant à l'objet. La seule bibliothèque externe que nous utilisons permet de modéliser les expressions régulières (formalisme décrit section 3.4.7).

Indépendance vis-à-vis du système d'exploitation

Tout le code de la bibliothèque correspond à du C++ ANSI/ISO. Nous utilisons la bibliothèque de modèles standards (STL pour *Standard Template Library* en anglais) pour nos conteneurs [15] ainsi que la bibliothèque REGEX++ [16] (disponible sur le site *Boost.org* [17]) pour les expressions régulières. Nous n'utilisons pas la bibliothèque de *Microsoft* (MFC pour *Microsoft Foundation Class*). Notre bibliothèque peut donc être compilée sur n'importe quelle plate-forme possédant un compilateur C++ avec les bibliothèques standards auxquelles il faut adjoindre la bibliothèque REGEX++.

3.4.14 Pistes de développement

Extension du formalisme des requêtes

Au niveau de la bibliothèque, et plus précisément au niveau du formalisme des *requêtes* et des *masques*, nous avons plusieurs pistes de développement.

La première consiste à étendre le formalisme des MER de manière à pouvoir gérer des corpus arborés. De tels formalismes existent déjà pour des applications comme TGREP (cf. section 3.3.7) ou encore TIGERSEARCH [18].

D'un autre côté, afin de nous décharger du problème de l'analyse syntaxique des *requêtes*, nous pensons utiliser une application comme GOLD PARSER [19]. La grammaire du formalisme des *requêtes* pourrait ainsi être complètement dissociée du code de la bibliothèque.

Lorsqu'une requête ne porte que sur un certain type de mots, les mots pleins par exemple, le filtrage doit être réalisé dans l'écriture même des MER alourdissant ainsi cette écriture. Il serait possible de spécifier un filtre permettant de préciser les mots à considérer. L'écriture de la MER et des masques s'en trouverait alors largement allégée. Un tel filtre pourrait prendre une forme analogue à celle de la clause *stop*.

Nous pensons également proposer un formalisme simplifié et allégé pour faciliter l'écriture de requêtes extrêmement simples.

Prise en compte de nouveaux formats de corpus

Dans le cadre du projet SYNTSEM, un marquage syntaxique peu profond faisant apparaître les constituants majeurs est en cours de réalisation. Nous comptons tenter d'exploiter ce marquage (cf. section 8.3). Le corpus comportant cette information syntaxique étant au format XML, il faudra développer une nouvelle classe corpus pour gérer ce format.

Actuellement, l'étiquetage du corpus (au niveau morphosyntaxique et lemmatisation) est effectué en amont par une application distincte. La société *Synapse Développement* proposant une bibliothèque C++ permettant de réaliser cet étiquetage, il serait possible de développer une classe qui puisse prendre en compte des corpus de texte brut

15. Dans le monde de la programmation orientée objet, les conteneurs sont des objets qui contiennent des objets. Ils permettent donc de modéliser des structures de données.

16. « Dr. John Maddock, Copyright (c) 1998-2001, version 3.31, 16 décembre 2001. Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Dr John Maddock makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty. »

17. Adresse Internet ; `http://www.boost.org/`

18. Site Internet : `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/` .

19. Site Internet : `http://www.devincook.com/goldparser/index.htm`.

en réalisant un étiquetage à la volée, de manière à pouvoir exploiter tout le pouvoir expressif des *requêtes*, même sur des corpus non étiquetés.

À court terme, nous pensons avoir besoin d'utiliser l'information de bases de données externes (pour les synonymes, les hyperonymes, etc.). Ce type d'information peut être traité au niveau d'une nouvelle classe corpus car la propriété [20] d'un mot, associée habituellement à une information se trouvant dans un corpus, peut très bien correspondre à une information située dans une base de données externe. Une classe corpus peut également appeler un programme effectuant un traitement particulier, comme une détermination thématique, pour répondre à une demande d'information correspondant à une propriété particulière dans une *requête*.

3.5 Applications (Win/DOS)LoX et CooLoX

3.5.1 Application (Win/DOS)LoX

L'application (Win/DOS)LoX a été développée autour de la bibliothèque LoX. Il s'agit d'un puissant outil de recherche et d'observation de phénomènes linguistiques dans les corpus écrits. Un phénomène linguistique est décrit formellement par une méta expression régulière (cf. définition 3.13). Les portions du corpus décrites par la méta expression régulière sont appelées des correspondances (cf. définition 3.1). Pour chacune des correspondances, l'application permet de générer diverses chaînes de caractères qui constituent le résultat du traitement. Différents masques (cf. définition 3.15) permettent de générer ces chaînes à partir des correspondances. Les masques possèdent un pouvoir expressif assez grand pour générer des phrases (pour une utilisation du style concordancier), des références (pour retrouver la correspondance dans le corpus), ou d'autres chaînes pouvant servir à un traitement automatique ou manuel subséquent.

(Win/DOS)LoX est en fait un acronyme pour désigner deux applications distinctes, WinLoX et DOSLoX, qui ont exactement les mêmes fonctionnalités. La seule différence entre les deux est que DOSLoX est orientée *ligne de commande* tandis que WinLoX, qui ne fonctionne que sous *Microsoft Windows*, possède une interface graphique. DOSLoX et WinLoX sont entièrement compatibles dans le sens où ils travaillent sur les mêmes fichiers et peuvent donc les échanger.

Nous avons recours à l'application (Win/DOS)LoX de manière récurrente tout au long de ce mémoire. Aussi est-il vivement recommandé de parcourir son manuel d'utilisation en annexe A. Nous y développons, entre autres, un exemple de requête complexe dans la section A.5.

3.5.2 Concordancier CooLoX

CooLoX est un puissant concordancier qui permet de tirer parti de tout le pouvoir expressif des méta-expressions régulières de la bibliothèque LoX, pour définir la cible ou filtrer les contextes droit et gauche. Étant amené à interagir fortement avec l'utilisateur, CooLoX possède une interface graphique. Aucune version orientée *ligne de commande* n'est disponible : elle n'aurait aucun sens puisque DOSLoX peut très bien produire des concordances (cf. section 4.4.1).

CooLoX comporte une fonctionnalité permettant de réaliser un étiquetage vertical dans de bonnes conditions. Ce module peut très bien être utilisé pour effectuer des corrections ou des modifications dans le corpus.

Le manuel d'utilisation de CooLoX se trouve en annexe B.

20. *Propriété* au sens défini dans le formalisme des requêtes, cf. section 3.4.5.

3.5.3 Améliorations futures des applications

De manière à ce que toutes nos applications soient portables, nous pensons réécrire (WIN/DOS)LOX et COOLOX en utilisant la bibliothèque graphique QT [21].

Nous projetons de développer notre concordancier COOLOX afin qu'il puisse fonctionner sur des corpus oraux transcrits et alignés *texte/son*. Il nous faudra donc développer une autre classe corpus pour gérer ce type de corpus. Cette amélioration ne consiste pas seulement à développer une nouvelle classe corpus mais, implique également une refonte partielle de l'interface graphique de l'application pour proposer les fonctionnalités allant de pair avec ce type de corpus (comme l'écoute de portions du corpus, par exemple).

Hormis le portage vers QT, l'application (WIN/DOS)LOX n'attend pas, elle, d'importants développements. En effet, il ne s'agit que d'une interface graphique rudimentaire destinée à faciliter la saisie de paramètres. Cette application permet essentiellement de tirer directement parti de toute la puissance de la bibliothèque. Toutes les améliorations dont bénéficiera l'application (WIN/DOS)LOX correspondront essentiellement à des améliorations apportées au niveau de la bibliothèque LOX.

21. La bibliothèque QT permet de développer des applications comportant une interface graphique capables d'être exécutées sous les environnements *Microsoft Windows*, *Unix/Linux* et *Mac Os X*. Le site Internet de QT est : `http://www.trolltech.com/` .

Chapitre 4

Dictionnaire, corpus et vocables

4.1 Introduction

L'une des difficultés majeures de l'étiquetage lexical automatique réside dans l'inadéquation des dictionnaires traditionnels (Véronis, 2001) ou dédiés (Palmer, 1998) pour cette tâche. Pour illustrer cette difficulté, nous pouvons citer le degré d'accord de seulement 57% (Ng & Lee, 1996) entre deux équipes indépendantes qui ont annoté un corpus en utilisant le dictionnaire électronique dédié WordNet. En utilisant un dictionnaire traditionnel, ce désaccord entre annotateurs peut devenir aussi important que si l'affectation des lexies avait été faite au hasard (Véronis, 1998). Pour pallier cette difficulté, notre équipe a entrepris la construction d'un dictionnaire distributionnel en se basant sur un ensemble de critères différentiels stricts (Reymond, 2002). Ce dictionnaire comporte pour l'instant la description détaillée de 20 noms communs, 20 verbes et 20 adjectifs. Nous discutons du problème de l'inadéquation des dictionnaires actuels ainsi que du dictionnaire distributionnel dans la **section 4.2**.

Une autre difficulté provient du manque de corpus lexicalement désambiguïsés sur lesquels des méthodes d'apprentissage supervisé peuvent être entraînées (Gale *et al.*, 1992b). Ce manque se transforme même en absence totale pour une langue comme le français (Kilgarriff, 1998a). Quelques corpus lexicalement désambiguïsés commencent toutefois à apparaître pour l'anglais, notamment dans le cadre des campagnes d'évaluation Senseval. Pour pallier cette difficulté, nous avons utilisé le dictionnaire distributionnel afin d'étiqueter manuellement chacune des occurrences des 60 vocables de notre étude dans le corpus du projet SyntSem (corpus d'environ six millions de mots, composé de textes de genres variés).

Dans la **section 4.3**, nous présentons la constitution et la préparation de ce corpus.

Dans la **section 4.4** nous détaillons la phase d'étiquetage lexical manuel ainsi que la finalisation du corpus.

Dans la **section 4.5**, nous donnons des informations quantitatives sur les 60 vocables de l'étude ainsi que sur le corpus du projet SyntSem.

Avant de poursuivre notre exposé, nous devons définir ce que nous entendons par *lexique*, *lexie* et *vocable*.

Définition 4.1 – Lexique –
C'est un ensemble très vaste des sens de base de la langue avec leur moyen d'expression associé.

Définition 4.2 – Lexie –
Nous appelons unité lexicale *ou* lexie *un élément de base du lexique. Chaque lexie de la langue est associée à un sens particulier, unique, définissable et à un ensemble de formes correspondant aux variantes flexionnelles du signifiant de la lexie.*

Définition 4.3 – Vocable –
Un vocable est un ensemble de lexies associées aux mêmes signifiants et dont les sens présentent une intersection non triviale.

4.2 Dictionnaire distributionnel

4.2.1 Notion de sens dans les dictionnaires classiques

La discrimination des sens constitue un problème majeur dans le cadre du traitement automatique des langues. Les lexicographes sont aussi directement confrontés à ce problème. Il n'est pas difficile de s'en convaincre en regardant les listes des sens données par deux dictionnaires différents pour un même mot : généralement, ces listes ne concordent pas. Bien que tout le monde s'accorde à dire que les mots peuvent avoir plusieurs sens, la précision de cette intuition reste un obstacle. Il n'existe en fait aucun fondement théorique au concept de sens d'un mot. Il est donc très difficile de dresser une liste des sens d'un mot de manière objective et systématique. La décision de subdiviser un sens en deux ou la décision de fusionner deux sens en un seul est souvent purement arbitraire. En réalité, la notion de sens semble être continue et non discrète. Cela explique pourquoi des dictionnaires différents donnent des listes de sens différentes pour un même mot. La question de la pertinence de l'utilisation de tels dictionnaires (LDOCE, COMLEX, ROGET, OALDCE, WORDNET, etc.) dans le TAL se pose alors. En fait, ces dictionnaires sont souvent inadaptés pour une telle tâche.

Les lexicographes ne conçoivent pas les dictionnaires dans l'optique de leur utilisation dans le domaine du TAL. Pour établir la définition d'un mot, et donc la liste des sens de ce mot, les lexicographes doivent respecter les contraintes suivantes :

- la présentation doit rester traditionnelle pour ne pas dérouter les lecteurs ;
- il faut tenir compte du fait que l'ouvrage est un ouvrage imprimé ; d'où la nécessité de produire des définitions compactes ;
- la méthode de l'accès au lexique doit être unique et simple ;
- la liste des sens d'un mot doit permettre de résoudre des conflits au sujet de la définition et du champ d'utilisation du mot.

Il faut donc garder à l'esprit que la principale contrainte à laquelle est soumis un lexicographe dans l'élaboration d'un dictionnaire est une contrainte commerciale. La population à laquelle s'adresse un dictionnaire est très vaste et ne se restreint pas aux linguistes. Son élaboration est par conséquent un enjeu économique important. Son but doit être de répondre aux besoins du plus grand nombre et pas spécialement aux besoins des linguistes. Or, à quoi un dictionnaire sert-il le plus souvent ? Le dictionnaire est, le plus souvent, utilisé soit pour résoudre des conflits humains, souvent familiaux, au sujet de la définition d'un mot, soit pour comprendre un mot jusqu'alors inconnu d'un individu. Pour cela, un dictionnaire doit avant tout être clair. Il doit délimiter sans ambiguïté et sans chevauchement les différents sens usuels d'un mot. Il apparaît donc clairement que les dictionnaires actuels ne peuvent répondre aux besoins du TAL.

4.2.2 Dictionnaire distributionnel et notion d'usage

Depuis la définition de Meillet (1926), selon laquelle le sens se définit par la moyenne de ses usages linguistiques (« Le sens d'un mot ne se laisse définir que par une moyenne entre ses emplois linguistiques »), de nombreuses théories sur la signification adoptent cette vision (Bloomfield, 1933 ; Hjemslev, 1953 ; Harris, 1954 ; etc.). Par exemple, le projet COBUILD (Sinclair, 1987) tente d'associer les sens du dictionnaire avec leurs usages en créant des divisions de sens basées sur des regroupements de citations du corpus. Dans la même optique, Schütze (1992, 1998) propose une approche dans laquelle chaque mot est représenté par un vecteur. Ces vecteurs sont automatiquement regroupés en paquets en fonction de leur degré de similitude, chaque paquet étant supposé être représentatif d'un sens.

C'est dans cette optique que s'inscrit la construction de notre dictionnaire distributionnel. L'information idéalement contenue dans ce dictionnaire n'a plus pour objectif premier la définition du sens comme dans les dictionnaires traditionnels, mais vise à organiser les vocables en lexies possédant des propriétés distributionnelles cohérentes.

La méthodologie adoptée s'inspire largement des idées mises en œuvre dans la construction du Dictionnaire Explicatif et Combinatoire (DEC) de Mel'cuk et son équipe (Mel'cuk, Clas & Polguere, 1995). Toutefois, à la différence de Mel'cuk qui utilise le corpus comme outil de vérification *a posteriori* après une construction lexicale introspective, nous considérons le corpus comme réservoir d'observations à partir desquelles les entrées doivent être élaborées, la démarche introspective étant réduite au minimum. L'entrée de dictionnaire, dans notre approche, est donc vue comme un modèle qui doit rendre compte au mieux des observations, et doit être prédictif des observations futures (dans d'autres corpus ou un corpus plus large). Le travail d'élaboration des entrées se fait de façon totalement interactive avec l'étiquetage d'un corpus de référence (le plus large possible), selon une stratégie incrémentale : le corpus est étiqueté (avec l'assistance d'outils informatiques comme CooLoX) au fur et à mesure de la construction des entrées, et les entrées sont révisées en fonction des nouveaux contextes rencontrés. À la fin de ce processus, nous obtenons un dictionnaire dont chacune des lexies est liée à un ensemble de contextes (*i.e.* un usage) dans le corpus. De même, chaque occurrence dans le corpus est étiquetée par le numéro d'une lexie dans le dictionnaire.

Les lexies de notre dictionnaire correspondent plus à des usages qu'à des sens. Par abus et commodité de langage, dans la suite de ce document, nous continuerons à utiliser le terme de sens.

4.3 Préparation et étiquetage automatique du corpus

4.3.1 Présentation du projet SyntSem

Des corpus du français étiquetés au niveau morphosyntaxique ainsi que des outils efficaces pour ce type d'annotation commencent à être disponibles. En revanche, ni étiquetage syntaxique, ni étiquetage lexical ne sont pour l'instant disponibles. Ce sont pourtant des ressources qui font cruellement défaut dans le domaine du TAL. Le projet SyntSem vise à corriger partiellement ce manque. Son objectif, extrêmement simple, est de fournir un corpus étiqueté au niveau morphosyntaxique intégrant deux types d'informations supplémentaires dont la conjonction est de première importance pour de nombreuses applications :

- un **marquage syntaxique peu profond** (*shallow parsing*), faisant apparaître les constituants majeurs ;

– un **marquage lexical** donnant le sens de mots sélectionnés pour leur caractère particulièrement polysémique.

La plupart des projets d'étiquetage de corpus réalisent un étiquetage total du corpus. Cet étiquetage exhaustif est intéressant pour certains types d'étiquetage, comme l'étiquetage morphosyntaxique, où la distribution des étiquettes est relativement uniforme, ce qui permet l'observation d'un nombre suffisant de cas relevant de chacune des catégories. En ce qui concerne les phénomènes syntaxiques et sémantiques, cette loi n'est pas vérifiée et l'étiquetage exhaustif ne permet pas d'obtenir un nombre suffisant d'observations pour chaque phénomène. Ainsi, par exemple, le corpus JOC-QE (questions écrites des parlementaires européens à la commission) d'un million de mots ne comporte environ qu'une centaine d'occurrences de mots polysémiques pourtant très courants tels que *biologique*, *clair*, *correct*, *barrage*, *chef*, *degré*, *arrêter*, *comprendre*, *tirer*, etc., et certains sens ou constructions syntaxiques de ces mots n'interviennent qu'une ou deux fois. Des corpus considérables sont donc nécessaires à l'étude des phénomènes syntaxico-sémantiques, et leur étiquetage exhaustif est pour l'instant d'un coût rédhibitoire, étant donné que l'étiquetage syntaxique (même peu profond) demande une très grande part de correction manuelle, et que l'étiquetage lexical ne peut pratiquement pas être automatisé pour l'instant. Cependant, dans la phase d'« amorçage » dans laquelle nous sommes, l'étiquetage syntaxico-sémantique peut être restreint aux seules phrases contenant des mots-cibles sélectionnés, puisque les relations syntaxico-sémantiques ne peuvent déborder le cadre de la phrase.

4.3.2 Sélection des vocables de l'étude

L'étiquetage lexical du projet SYNTSEM porte sur 60 mots-cibles (20 adjectifs, 20 noms, 20 verbes), qui ont été sélectionnés selon une méthodologie rigoureuse dans le cadre de l'action d'évaluation SENSEVAL-ROMANSEVAL (Véronis, 1998) de façon à obtenir des mots fortement polysémiques sans biais préalable (tel que la consultation d'un dictionnaire particulier, ou l'intuition). 600 mots (200 adjectifs, 200 noms, 200 verbes) ont tout d'abord été extraits du corpus JOC-QE selon des critères de fréquence. Les occurrences de ces mots dans l'ensemble du corpus ont été soumises sous forme de lignes de concordance regroupant tous les contextes d'un mot sur une page (600 pages au total). Six étudiants ont été rémunérés pour examiner chaque page et répondre à la question : « Le mot x a-t-il plusieurs sens dans les contextes ci-dessous ? ». Les 20 mots de chaque catégorie, adjectif, nom, verbe (60 mots au total) qui faisaient l'objet du plus grand nombre de réponses positives ont été retenus comme mots-cibles. Ces mots sont tous à fort caractère polysémique. Le tableau 4.1 donne la liste des 60 vocables sélectionnés.

4.3.3 Constitution du corpus

Le corpus du projet SYNTSEM est une compilation d'un ensemble de textes de genres relativement variés. Le corpus a été réalisé en 1999-2000 par Benoît Habert et Jean Véronis, dans le cadre d'un contrat avec ELRA/ELDA, à partir de sources provenant notamment des projets PAROLE et MULTEXT. Ce corpus est constitué de cinq sous-corpus d'environ un million de mots chacun.

ABU : Le corpus ABU est une compilation de textes intégraux d'œuvres littéraires (fin XIX^{e} début XX^{e}) du domaine public francophone produits et diffusés par les membres bénévoles de l'Association des Bibliophiles Universels [1]. Il est constitué

1. Site Internet : `http://abu.cnam.fr/` .

NOMS	ADJECTIFS	VERBES
barrage	biologique	arrêter
chef	clair	comprendre
communication	correct	conclure
compagnie	courant	conduire
concentration	exceptionnel	connaître
constitution	frais	couvrir
degré	haut	entrer
détention	historique	exercer
économie	plein	importer
formation	populaire	mettre
lancement	régulier	ouvrir
observation	sain	parvenir
organe	secondaire	passer
passage	sensible	porter
pied	simple	poursuivre
restauration	strict	présenter
solution	sûr	rendre
station	traditionnel	répondre
suspension	utile	tirer
vol	vaste	venir

Tableau 4.1 – Liste des 60 vocables sélectionnés pour l'étude.

des œuvres suivantes :

- « La légende et les aventures héroïques, joyeuses et glorieuses d'Ulenspiegel » de Charles de Coster (1867) ;
- « La théorie physique, son objet, sa structure » de Pierre Duhem (1906) ;
- « Bouvard et Pécuchet » de Gustave Flaubert (1881) ;
- « Quatre-vingt-treize » de Victor Hugo (1874) ;
- « Les preuves » de Jean Jaurès (1898) ;
- « Bel Ami » de Guy de Maupassant (1885) ;
- « Journal de Jules Renard » de Jules Renard (1894-1904) ;
- « La curée » de Émile Zola (1872).

JOC : Le corpus JOC est constitué de textes du Journal Officiel de la Commission Européenne [2] (1993, Série C, Questions et réponses des parlementaires européens à la Commission).

MON : Le corpus MON est une compilation d'extraits du journal *Le Monde* [3] (des articles tirés au hasard dans les années 1987, 1989, 1991, 1993 et 1995).

OUV : Le corpus OUV est une compilation d'ouvrages en sciences humaines provenant des éditions du CNRS :

- « Évolutionnisme et fixisme en France Histoire d'un combat » (1800-1882) ;
- « La conversion des intellectuels au catholicisme en France » (1885-1935) ;
- « Cinquante ans de traction à la SNCF » ;
- « La protection des œuvres scientifiques en droit d'auteur français » ;
- « Le darwinisme social en France » (1859-1918) ;

2. Site Internet : `http://www.lpl.univ-aix.fr/projects/multext/CORP/JOC.html` .
3. Copyright : LE MONDE S.A. Paris, France.

Corpus	*Paragraphe*	*Position*	*Jeton*
M	1	1	La
M	1	2	première
M	1	3	réunion
M	1	4	des
M	1	5	douze
M	1	6	hauts
M	1	7	fonctionnaires
M	1	8	chargés
M	1	9	de
M	1	10	coordonner
M	1	11	l
M	1	12	’
M	1	13	opération
M	1	14	consistant
M	1	15	à
M	1	16	supprimer
M	1	17	les
M	1	18	frontières

Figure 4.1 – Extrait du corpus segmenté avec références.

- « L’efficience productive » ;
- « Emploi et régulation » ;
- « Progrès scientifique et responsabilité administrative » ;
- « La publicité. Naissance d’une profession » ;
- « Les procureurs du droit ».

PER : Le corpus PER est une compilation d’articles issus de périodiques : des articles longs (10 000 à 20 000 mots) en communication politique (revue Hermès) et des communiqués courts (1000 à 2000 mots) de vulgarisation des résultats de la recherche (revue CNRS INFO).

4.3.4 Segmentation et référence

L’objectif de la segmentation est de découper chaque phrase en un certain nombre de jetons. Pour réaliser cette segmentation, nous utilisons le principe de la segmentation maximale qui consiste à réaliser une segmentation sur tous les caractères non alphanumériques.

Chaque jeton reçoit une référence unique de manière à rendre possible la synchronisation entre les différentes versions du corpus et entre les différents projets. Cette référence se subdivise en trois informations. La première est une lettre (*A*, *J*, *M*, *O*, *P*) qui permet d’identifier le sous-corpus (*ABU*, *JOC*, *MON*, *OUV*, *PER*). La seconde est le numéro du paragraphe à l’intérieur du sous-corpus. La dernière est la position du jeton dans le paragraphe. La figure 4.1 représente un extrait du corpus segmenté avec référence.

4.3.5 Étiquetage morphosyntaxique et lemmatisation

La lemmatisation et l’étiquetage morphosyntaxique sont effectués de manière automatique avec le logiciel CORDIAL ANALYSEUR 9 dont le taux d’étiquettes correctes est

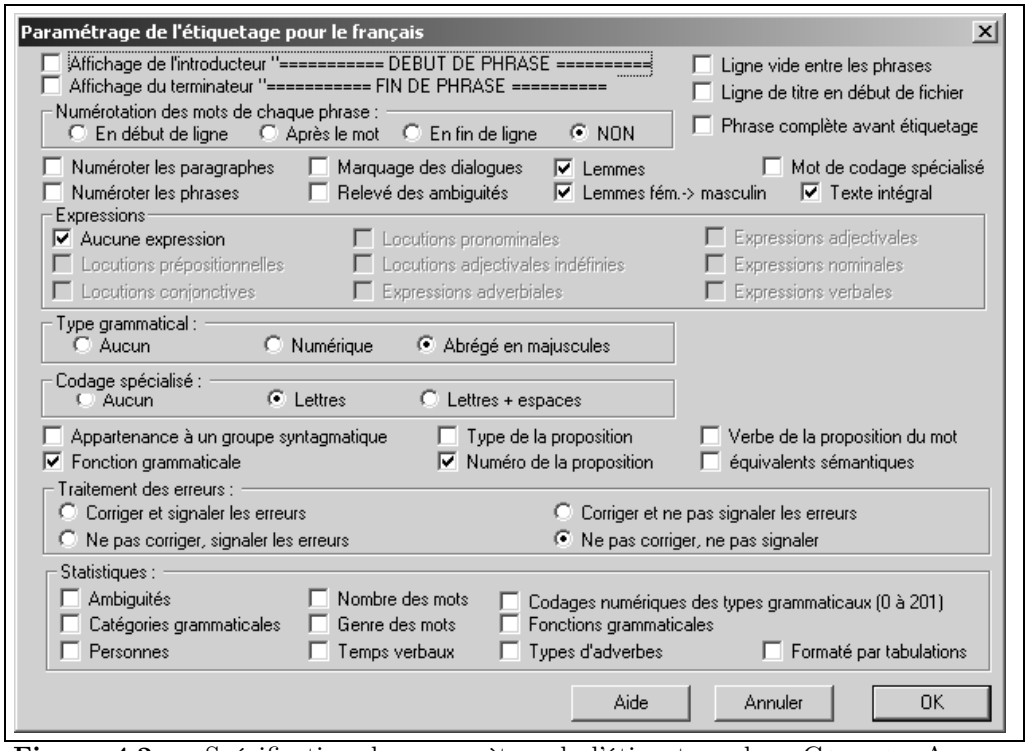


Figure 4.2 – Spécification des paramètres de l'étiquetage dans CORDIAL ANALYSEUR 9.

de l'ordre de 98% toutes catégories confondues (Valli & Véronis, 1999).

Cet étiquetage se fait à partir des cinq fichiers (un par sous-corpus : *ABU*, *JOC*, *MON*, *OUV*, *PER*) en version texte brut. En effet, ce format est le seul accepté par le logiciel CORDIAL ANALYSEUR qui n'accepte pas notre corpus au format segmenté avec référence.

Pour étiqueter un texte avec le logiciel CORDIAL ANALYSEUR, il suffit d'ouvrir celui-ci et de choisir l'item *Étiquetage de texte* dans le menu *Syntaxe*. Cette action ouvre une boîte de dialogue qui permet de spécifier tous les paramètres de l'étiquetage que l'utilisateur désire réaliser. Nous ne détaillons pas ces paramètres. La figure 4.2 représente une copie d'écran de la boîte avec les paramètres spécifiés pour réaliser l'étiquetage de nos sous-corpus.

À l'issu de cet étiquetage le corpus comporte six colonnes :

- jeton ;
- lemme ;
- étiquette de type CORDIAL ANALYSEUR ;
- étiquette de type MULTEXT-GRACE ;
- fonction grammaticale du mot ;
- numéro de la proposition.

L'ensemble des étiquettes de type CORDIAL ANALYSEUR et MULTEXT-GRACE, présentes dans le corpus, sont répertoriées et dénombrées en annexe section E. La signification des étiquettes de type CORDIAL ANALYSEUR est détaillée section E.3 et celle du projet MULTEXT-GRACE en section E.4.

jeton	*lemme*	*étiq. type* CORD. ANA.	*étiq. type* MULT.-GRA.	*fonc. gram.*	*num. prop.*
La	le	DETDFS	Da-fs-d	T	1
première	premier	ADJORD	Ao-..	T	1
réunion	réunion	NCFS	Ncfs	T	1
des	de	DETDPIG	Da-.p-i	T	1
douze	douze	ADJNUM	Mc.p	T	1
hauts	haut	ADJMP	Afpmp	T	1
fonctionnaires	fonctionnaire	NCPIG	Nc.p	T	1
chargés	charger	VPARPMP	Vmpapm	T	1
de	de	PREP	Sp	T	1
coordonner	coordonner	VINF	Vmn--	T	1
l'	le	DETDFS	Da-ms-d	D	1
opération	opération	NCFS	Ncfs	D	1
consistant	consister	VPARPRES	Vmpp--	-	1
à	à	PREP	Sp	F	1
supprimer	supprimer	VINF	Vmn--	F	1
les	le	DETDPIG	Da-.p-d	N	1
frontières	frontière	NCFP	Ncfp	N	1

Figure 4.3 – Extrait du corpus après étiquetage morphosyntaxique et lemmatisation.

La colonne *fonction grammaticale du mot* fournit la fonction grammaticale du mot lorsque celle-ci est clairement identifiée. Ce codage sur une lettre est explicité en annexe section E.5.

La colonne *numéro de la proposition* donne le numéro de la proposition à laquelle appartient le jeton. Ce numéro est 1 pour la première proposition, 2 pour la suivante, etc. Une proposition est normalement constituée autour d'un verbe, mais elle peut aussi se constituer autour d'un présentatif comme *voici* ou *voilà*.

La figure 4.3 représente un extrait du corpus segmenté après étiquetage morphosyntaxique et lemmatisation.

L'information véhiculée par les étiquettes introduites dans cette section est primordiale pour la désambiguïsation lexicale (cf. section 2.7.2). La lemmatisation ou les étiquettes morphosyntaxiques permettent de résoudre les ambiguïtés grammaticales d'un jeton. Par exemple, l'adjectif *haut* fait partie de notre liste de vocables. Lorsque nous rencontrons le jeton *haut* nous n'avons pas à lever l'ambiguïté sur la classe grammaticale du jeton (adjectif ou nom), cette ambiguïté est levée par l'étiquetage que nous venons de réaliser. La lemmatisation permet également d'accéder immédiatement à toutes les formes fléchies de nos vocables. Nous pouvons, par exemple, accéder directement à toutes les formes fléchies de l'adjectif *haut*. Le lemme des jetons est une information très largement utilisée par un grand nombre de méthodes de désambiguïsation lexicale et permet, dans une certaine mesure, de répondre au problème de la dispersion des données (cf. section 2.5.6). Les étiquettes morphosyntaxiques peuvent également être utilisées (cf. section 2.7.2).

Bien évidemment, toutes ces étiquettes ne sont pas forcément correctes. Ainsi, dans tous les traitements subséquents qui tirent parti de cet étiquetage, nous sommes tributaire des erreurs d'étiquetage faites par CORDIAL ANALYSEUR, ou causées par la synchronisation du corpus étiqueté (cf. section 4.3.6). L'une des familles d'étiquettes la plus utilisée pour la désambiguïsation automatique est celle des lemmes. Ces étiquettes seront également régulièrement utilisées pour accéder automatiquement à toutes les formes fléchies de nos 60 vocables. Pour cette raison, nous nous efforcerons, par des

méthodes automatiques et peu coûteuses, de réduire les incohérences au niveau des lemmatisations (cf. section 4.3.7). Nous chercherons également à nous assurer que tous les vocables qui reçoivent une étiquette lexicale soient correctement lemmatisés (cf. section 4.4.2).

4.3.6 Synchronisation du corpus étiqueté

L'étape précédente d'étiquetage nous a fait perdre la synchronisation avec notre corpus segmenté. L'objectif de cette synchronisation est de restaurer les références dans le corpus. À l'issue de cette synchronisation le corpus doit comporter neuf colonnes :

- lettre identifiant le sous-corpus ;
- numéro de paragraphe ;
- position du jeton dans le paragraphe ;
- jeton ;
- lemme ;
- étiquette de type Cordial Analyseur ;
- étiquette de type Multext-Grace ;
- fonction grammaticale du mot ;
- numéro de la proposition.

Si la segmentation effectuée par Cordial Analyseur correspondait à la segmentation décrite section 4.3.4, la synchronisation deviendrait triviale. Hélas ce n'est pas le cas. De plus, les multiples ajouts et omissions de Cordial Analyseur par rapport au texte de départ rendent cette tâche délicate.

La désynchronisation la plus classique provient d'un découpage en jetons plus important au niveau de notre segmentation qu'au niveau de celle faite par Cordial Analyseur. Par exemple, nous segmentons la phrase *C'est-à-dire celles qui sont à Lille* de la manière suivante :

C | ' | est | - | à | - | dire | celles | qui | sont | à | Lille

Pour cette même phrase, Cordial Analyseur effectue la segmentation qui suit :

C'est-à-dire | celles | qui | sont | à | Lille

Le cas inverse, bien que plus rare, peut également se produire. Par exemple, pour la phrase *fabrication de circuits imprimés et de câbles (A2E)* notre segmentation donne :

fabrication | de | circuits | imprimés | et | de | câbles | (| A2E |)

Pour cette même phrase, Cordial Analyseur effectue la segmentation qui suit :

fabrication | de | circuits | imprimés | et | de | câbles | (| A2 | E |)

Vient ensuite toute une série de problèmes dont l'origine semble être des *coquilles* du le logiciel Cordial Analyseur. Il y a tout d'abord certains nombres écrits en chiffres qui sont parfois ignorés. Cette désynchronisation peut devenir très importante lorsque le texte original contient un tableau de chiffres. Il y a ensuite des ajouts de Cordial Analyseur, qui correspondent souvent à des corrections spontanées, comme la phrase *au stade où on se dit* segmentée en :

au | stade | où | l' | on | se | dit

Il y a enfin de nombreuses lignes comportant la seule chaîne `\r`, des jetons commençant par un espace et des jetons contenant un caractère 13 (retour chariot) suivis d'une portion de la suite du texte.

Pour résoudre tous ces problèmes et obtenir un corpus étiqueté synchronisé avec notre segmentation, nous avons développé en `C++` une petite application entièrement

```
A  1885  45  des         un          DETDPIG  Da-.p-i  H  1
A  1885  46  cailloux    caillou     NCMP     Ncmp     H  1
A  1885  47  qui         qui         PRI      Pr-..n   S  2
A  1885  48  allaient    aller       VINDI3P  Vmii3p   V  2
A  1885  49  eux         lui-même    PPER3P   Pp3mpd   H  3
A  1885  50  -           lui-même    PPER3P   Pp3mpd   H  3
A  1885  51  mêmes       lui-même    PPER3P   Pp3mpd   H  3
A  1885  52  redevenir   redevenir   VINF     Vmn--    V  3
A  1885  53  les         le          DETDPIG  Da-.p-d  -  3
A  1885  54  rochers     rocher      NCMP     Ncmp     -  3
```

Figure 4.4 – Extrait du corpus étiqueté et synchronisé.

dédiée à cette synchronisation. Cette application permet de resynchroniser automatiquement et de manière efficace le corpus étiqueté.

La figure 4.4 représente un extrait du corpus étiqueté et synchronisé.

4.3.7 Décomposition des lemmatisations regroupées

Nous avons vu que notre découpage en jetons était plus drastique que celui de Cordial Analyseur. Ainsi, par rapport à notre découpage, Cordial Analyseur a effectué un grand nombre de regroupements. Comme le montre la figure 4.4 avec le regroupement *eux-mêmes*, tous les jetons qui constituent ce regroupement reçoivent le lemme *lui-même*. Ainsi, ces trois lemmes *lui-même* ne correspondent pas aux jetons auxquels ils sont associés mais sont des traces d'un regroupement effectué par Cordial Analyseur. Ce constat n'est pas très satisfaisant, d'autant plus que ces regroupements sont fréquents et contiennent parfois l'un des 60 vocables de notre étude (*chef-d'œuvre*, *couvre-chef*, *contre-pied*, *chauffe-pied*, *sous-station*, etc.). Il faut impérativement que le lemme des occurrences de nos 60 vocables soit correct de manière à pouvoir accéder aux occurrences de ces vocables par leur lemme.

Nous considérons qu'un lemme correspond à un regroupement si et seulement si :

- il contient un tiret (`-`), un point (`.`), une apostrophe (`'`) ou une espace ;
- il est composé de deux caractères ou plus ;
- il est identique au lemme qui le précède ou qui lui succède.

WinLoX nous permet de facilement repérer et dénombrer les différentes formes que prennent les lemmes de chacun de ces regroupements. Nous avons ainsi dénombré 246 115 lemmes correspondant à un regroupement. La figure 4.5 donne la liste des lemmes les plus fréquents qui correspondent à des regroupements.

Pour réduire l'impact, au niveau des lemmes, de ces regroupements, nous avons développé en `C++` une petite application entièrement dédiée à la décomposition des lemmes qui correspondent à des regroupements. D'autre part, cette application corrige systématiquement les jetons «'», «-» et «.» qui ne reçoivent pas un lemme identique au jeton. Toutes les autres étiquettes sont laissées inchangées lors de cette opération.

La décomposition des lemmatisations regroupées que nous venons de faire est loin d'être parfaite. En effet, certains lemmes sont erronés, par exemple des jetons *d* dont le lemme était *d'autres* vont se voir affecter le lemme *d*. D'autre part, des étiquettes morphosyntaxiques restent erronées, par exemple, le jeton apostrophe de *d'autres* conserve l'étiquette morphosyntaxique affectée par Cordial Analyseur au regroupement *d'autres*. Cependant, le corpus après décomposition des lemmes qui correspondent à des regroupements est tout de même bien plus cohérent en raison de la

m.	18000	Etats-Unis	1221	j.	502
--	11849	c.m..	1092	Udf-Cds	501
c'est-à-dire	8859	au-dessus	990	a priori	488
...	8764	Pays-Bas	969	réf.	484
d'autres	8268	pp.	932	d'ores et déjà	475
celui-ci	6216	e.	908	f.	472
aujourd'hui	5263	div.	906	près de	455
l'un	4054	c.r.	880	à l'instar	444
parce que	3973	Geoffroy sai...	832	Grande-Bretagne	438
lui-même	3903	c.g..	756	Jean-Claude	438
d'abord	3606	Jean-Pierre	747	jusque-là	414
peut-être	3300	quelqu'un	747	porte-parole	384
afin de	3067	celui-là	726	etc.	356
d.s.	2292	Udf-Pr	678	savoir-faire	351
l'une	1977	b.	642	rendez-vous	330
elle-même	1908	est-ce qu	635	quelques-uns	327
a.	1882	afin que	624	soixante-dix	324
États-Unis	1488	cnrs-université	591	vingt-cinq	324
la plupart des	1443	ci-dessus	574	dix-huit	321
vis-à-vis	1425	v.	558	ex-yougoslavie	312
est-ce que	1412	i.	548	là-bas	312
tandis que	1407	p.	540	chef-d'oeuvre	310
Royaume-Uni	1311	n.	522	Jean-Louis	306
à l'encontre	1284	l.	511	c.	305
au-delà	1242	g.	510	Jean-Paul	294

Figure 4.5 – Lemme des regroupements de CORDIAL ANALYSEUR les plus fréquents dans le corpus.

forte diminution des lemmes aberrants. Certains lemmes restent aberrants, mais ils le sont tout de même moins (il est préférable que le jeton *d* reçoive le lemme *d* que *d'autre*) et correspondent à des cas relativement isolés dans le corpus. Corriger toutes les aberrations au niveau des lemmes et des autres étiquettes nécessite un investissement disproportionné par rapport au gain que nous pourrions en tirer. D'autant plus qu'une grande partie des aberrations au niveau des lemmes et des autres étiquettes provient des erreurs d'étiquetage de CORDIAL ANALYSEUR. En effet, le taux d'étiquettes correctes de CORDIAL ANALYSEUR étant de l'ordre de 98%, la longueur moyenne d'une phrase étant de 26 mots (cf. section 4.5.1), CORDIAL ANALYSEUR génère en moyenne une erreur d'étiquetage toutes les deux phrases.

4.4 Étiquetage lexical manuel et finalisation du corpus

4.4.1 Étiquetage lexical manuel

La stratégie consistant à étiqueter chacune des occurrences des 60 vocables dans l'ordre linéaire du corpus est fastidieuse et conduit à de mauvais résultats. L'expérience de notre équipe nous a conduit à opter pour une stratégie d'étiquetage *vertical* (Reymond, 2001). Ainsi, l'étiquetage se fait par vocable à l'aide d'une présentation sous forme de concordances. Cette présentation est composée de quatre colonnes principales :

- contexte gauche de l'occurrence ;
- occurrence du vocable ;
- colonne vide destinée à recevoir l'étiquette lexicale de l'occurrence ;
- contexte droit de l'occurrence ;

Cette stratégie permet une meilleure cohérence des décisions d'étiquetage sur l'ensemble du corpus, et facilite grandement la tâche de l'annotateur en lui permettant d'utiliser des fonctions de tri, de sélections diverses, et d'étiquetage par paquets entiers d'occurrences.

Il faut remarquer que certains mots ont un lemme erroné qui correspond à celui de l'un de nos 60 vocables. Ces mots ne reçoivent pas une étiquette lexicale valide (c'est-à-dire commençant par un chiffre), mais une étiquette expliquant l'erreur de lemmatisation (par exemple l'étiquette `N` pour un nom ayant été lemmatisé comme l'un de nos 20 verbes).

L'application COOLOX aurait été tout à fait adaptée à cette phase d'étiquetage manuel (Reymond, 2002). Hélas, le développement de cette application est survenu après cette phase d'étiquetage. Nous avons donc utilisé l'application WINLOX pour générer des fichiers tabulaires qui sont ensuite édités par l'annotateur à l'aide d'un tableur (*Microsoft Excel* en l'occurrence). 60 fichiers sont ainsi créés : un par vocable. De manière à pouvoir ré-injecter les étiquettes lexicales dans notre corpus, il faut ajouter les trois colonnes de référence. Une huitième colonne est également ajoutée de manière à servir de clef de tri selon le contexte gauche. Les fichiers tabulaires générés pour chaque vocable par l'application WINLOX comportent ainsi les huit colonnes suivantes :

- lettre désignant l'initiale du sous-corpus de l'occurrence;
- numéro de paragraphe de l'occurrence ;
- position de l'occurrence dans le paragraphe ;
- contexte gauche de l'occurrence ;
- occurrence du vocable ;
- colonne vide destinée à recevoir l'étiquette lexicale ;
- contexte droit de l'occurrence ;

34	J	39797	29	e au contrôle de l ' acquisition et de la	détention	2	d ' armes ; 2) quels États membres appli	la de et acquisitio
35	J	5793	27	e munitions est identique à celui de la	détention	2	d ' armes à feu auquel elles sont destinée	la de celui à ident
36	M	10303	82	en inquiète un membre du comité . La	détention	2	d ' armes a l ' air chose banale . L ' avocat	La . comité du m
37	J	27242	23	e au contrôle de l ' acquisition et de la	détention	2	d ' armes En vertu du règlement de 1992 r	la de et acquisitio
38	J	10844	23	e au contrôle de l ' acquisition et de la	détention	2	d ' armes N ° 3301 / 92 de M . Ernest Glin	la de et acquisitio
39	J	5792	14	ective 91 / 477 / CEE (1) relative à la	détention	2	d ' armes ne contient pas de dispositions	la à relative) 1
40	M	13461	23	nvier (" Métronome ") craignait que la	détention	2	d ' armes ne soit hissée , en France , à la	la que craignait)
41	M	5743	87	en France , pour abus de confiance et	détention	2	d ' armes notamment . Les services spéci	et confiance de a
42	O	4648	94	évaluation qui peut ne pas aboutir à la	détention	2	d ' un diplôme . L ' accord établit claireme	la à aboutir pas n
43	J	27244	66	en possession d ' une autorisation de	détention	2	d ' un fusil de chasse ou d ' une arme à fe	de autorisation u
44	P	7118	216	e la compétence qui lui est reconnue (	détention	2	d ' un savoir et d ' un savoir - faire éprouvé	(reconnue est lu
45	J	5794	11	stinées » . L ' article 12 parle de « . . .	détention	2	d ' une arme à feu pendant un voyage . . »	. . . « de
46	J	5793	18	e que « le régime d ' acquisition et de	détention	2	de munitions est identique à celui de la dé	de et acquisition
47	J	5799	40	nes , la Commission considère que la	détention	2	de munitions pendant un voyage intracom	la que considère
48	J	5799	53	utaires obéit aux mêmes règles que la	détention	2	des armes auxquelles les munitions sont	la que règles mê
49	J	8571	16	unautaire , les conditions régissant la	détention	2	des lévriers et leur participation à des cou	la régissant cond
50	J	5799	104	plémentaire n ' est nécessaire pour la	détention	2	des munitions correspondant à cette arme	la pour nécessair
51	J	9394	88	on des procédures d ' inspection et de	détention	2	des navires . Selon les informations dont l	de et inspection '
52	J	10416	7	Objet : Amélioration des conditions de	détention	1	des prisonniers en Syrie Les ministres de	de conditions des
53	J	31012	56	possèdent - ils des informations sur la	détention	1	des prisonniers politiques , notamment du	la sur information
54	J	18786	6	1992 , p . 29 . Objet : Poursuite de la	détention	1	des prisonniers politiques au Viêt - nam L	la de Poursuite :
55	J	40067	42	t maintenus depuis plusieurs mois en	détention	1	préventive dans la prison de Korydallos . E	en mois plusieurs
56	J	10728	3	la Commission (6 . 1 . 1993) Objet :	Détention	1	préventive de l ' éditeur du journal grec « A	: Objet) 1993 .
57	J	23041	3	JO n ° L 40 du 11 . 2 . 1989 . Objet :	Détention	1	préventive de l ' éditeur du journal grec AV	: Objet . 1989 .
58	J	744	34	he Dev - Sol Guiutsler . Elles sont en	détention	1	préventive depuis le 16 janvier 1992 sans q	en sont Elles . G
59	J	23042	17	outsas , éditeur d ' Avgi , a été mis en	détention	1	préventive en raison des dettes contractée	en mis été a ,
60	J	20035	48	ernant la diminution de la durée de la	détention	1	préventive et les garanties quant à l ' unifo	la de durée la de
61	J	42843	19	, M . Mahmut Dogan avait été mis en	détention	1	préventive et relâché après 8 jours de garc	en mis été avait D
62	J	20034	9	. Objet : Diminution de la durée de la	détention	1	préventive Le Conseil juge - t - il opportun	la de durée la de

Figure 4.6 – Présentation d'un étiquetage vertical.

– clef de tri selon le contexte gauche de l'occurrence.

La figure 4.6 montre la présentation d'un tel fichier tabulaire.

La figure 4.7 montre la règle utilisée pour la réalisation des fichiers tabulaires et la figure 4.8 donne les paramètres du traitement pour la réalisation du fichier tabulaire pour le vocable *détention*. Grâce à l'utilisation de scripts (en utilisant donc DOSLoX), nous avons entièrement automatisé la réalisation de ces 60 fichiers tabulaires.

À ce stade, notre corpus comporte 10 colonnes identifiées par le fichier propriété de la manière suivante : *fichier*, *paragraphe*, *position*, *jeton*, *lemme*, *ems*, *tgb*, *fonc*, *prop* et *lexie*. Ce sont ces noms que nous utilisons pour désigner les étiquettes de colonnes particulières dans les méta-expressions régulières et les masques.

4.4.2 Occurrences non détectées

Comme nous pouvons le voir sur la figure 4.7 les occurrences des 60 vocables sont identifiées par leur lemme. Cette procédure permet d'accéder à toutes les formes fléchies d'un vocable en s'affranchissant des ambiguïtés catégorielles et en simplifiant au maximum la syntaxe de la requête qui décrit toutes les formes fléchies du vocable. Cet avantage comporte néanmoins un inconvénient : il nous rend tributaire de la qualité de la lemmatisation des occurrences de nos vocables dans le corpus. En effet, seules les occurrences correctement lemmatisées reçoivent une étiquette lexicale. Or, comme le montre la figure 4.9, certaines occurrences ne sont pas correctement lemmatisées et ne reçoivent donc pas d'étiquette lexicale.

Les erreurs au niveau de la lemmatisation peuvent avoir deux origines :

– une erreur de lemmatisation de la part du logiciel Cordial Analyseur ;

– une erreur de lemmatisation résultant de la décomposition des lemmatisations regroupées (opération décrite section 4.3.7).

```
Requête :
cible:[lemme="@(1)"]

Apparence :
// Fichier
[P:cible.fichier][C:9]
// Paragraphe
[P:cible.paragraphe][C:9]
// Position dans le paragraphe
[P:cible.position][C:9]
// Contexte gauche
[B:[P:mot][C:32];cible.index-20;cible.index-1][C:9]
// Cible
[P:cible.mot][C:9]
// Lexie
[P:cible.lexie][C:9]
// Contexte droit
[B:[P:mot][C:32];cible.index+1;cible.index+20][C:9]
// Clef contexte gauche
[B:[P:mot][C:32];cible.index-1;cible.index-5][C:9]
// Fin de ligne
[C:10]
```

Figure 4.7 – Règle (`etiq_vert.loxr`) utilisée pour la réalisation des fichiers tabulaires servants à l'étiquetage vertical dans un tableur.

```
\act=EXT_CORR
\opt_act=nodiscr
\opt_act=corp_ens
\corpus_type=tabulaire
\fic_prop=propriete.txt
\sep_prop=9
\sep_mot=10
\corpus=abu.cnr
\corpus=joc.cnr
\corpus=mon.cnr
\corpus=ouv.cnr
\corpus=per.cnr
\regle=etiq_vert.loxr<détention>
\fic_res=res.loxf
\fic_ref=
\fic_inf=inf.loxf
```

Figure 4.8 – Paramètres du traitement pour la réalisation du fichier tabulaire pour le vocable *détention*.

Contexte gauche	Jeton	Lemme	Contexte droit
réglé , les costumes	arrêtés	costumer	, les poses apprises ,
du service universel	compris	livrer	comme l ' ensemble des
hypothèses une fois	connues	être	, une déduction
, la plupart bien	connues	avoir	, bien souvent étudiées
sont imculpés , est	exceptionnelle	exceptionnelle	: 56 kilogrammes d '
) Objet : Contrôles	exercés	grecquer	cette année sur la
les « glen » des	hautes	haute	terres d ' Écosse ,
les modestes subsides	mis	justifier	initialement à sa
la bouche , les jambes	ouvertes	boucher	, le haut du pantalon
' arbre , les paumes	ouvertes	contrer	, il dormait , en
: - - La nuit	passée	nuire	, nous avons couché dans
, il y mordit à	pleines	pleine	dents . « Ah !
de leur architecture	régulière	régulière	les courbes étudiées des
journée du 18 janvier	rendrent	rendrent	compte des réactions et
sanguins Le compte	rendu	compter	officiel du Conseil des
les mutilations	traditionnelles	traditionnelle	sont prohibées par l '
, et , le soir	venu	pouvoir	, il résolut de passer
détails . La nuit	venue	nuire	, le Victoria jeta l

Figure 4.9 – Exemples d'occurrences mal lemmatisées.

Il est difficile de garantir que chacune des occurrences des 60 vocables est bien lemmatisée. En effet, même en regardant manuellement toutes les formes fléchies possibles de chacun de nos vocables, il subsisterait des erreurs de lemmatisation causées par une mauvaise interprétation de l'annotateur (erreurs humaines). D'autre part, une telle opération serait coûteuse en temps et fastidieuse. Nous avons donc simplement cherché à réduire ces erreurs en utilisant des méthodes automatiques et peu coûteuses.

Ainsi, pour détecter des occurrences de nos vocables comportant un lemme erroné, nous effectuons des recherches centrées sur les jetons et non pas sur leur lemmatisation. Nous utilisons parfois l'information véhiculée par les étiquettes morphosyntaxiques pour filtrer les résultats afin de ne pas surcharger l'annotateur en occurrences correctement lemmatisées correspondant à une autre classe grammaticale. Ce filtrage, quand nous y avons recours, se fait au détriment de l'exhaustivité de la détection puisque nous sommes cette fois-ci tributaire de la correction de l'étiquetage morphosyntaxique. Le principe de base de la détection consiste à rechercher des jetons dont la forme correspond à une forme fléchie de l'un de nos vocables mais dont le lemme ne correspond pas à celui de ce vocable. Dans le cas des adjectifs, nous excluons les jetons reconnus comme des noms, pour limiter le nombre de détections inadéquates. Pour les verbes, nous nous limitons aux jetons reconnus comme des verbes. Cette procédure permet d'obtenir une bonne précision de la détection au détriment du rappel. Nous détectons ainsi environ un millier d'occurrences des 60 vocables potentiellement mal lemmatisées parmi lesquelles environ 90% le sont vraiment. Ces recherches sont effectuées avec le logiciel WinLoX qui produit comme résultat un fichier tabulaire, comme décrit dans la section 4.4.1, utilisé pour réaliser un étiquetage lexical manuel. À titre d'exemple, la figure 4.10 montre la règle utilisée pour la détection des occurrences des 20 noms potentiellement mal lemmatisés, la figure 4.11 montre la règle utilisée pour les adjectifs. Pour utiliser ces règles, l'action de WinLoX est *extraire correspondances*.

Notre travail s'étalant sur plusieurs années, nous avons disposé de plusieurs mises-à-jour du logiciel Cordial Analyseur. Nous avons ainsi repris plusieurs fois la procédure d'étiquetage de notre corpus ; bien entendu, seul l'étiquetage automatique est recommencé, l'étiquetage lexical manuel est toujours conservé. En fait, avec la dernière version utilisée, notre procédure de détection des occurrences des 60 vocables potentiellement mal lemmatisés ne détecte plus qu'une centaine d'occurrences potentiellement

```
Requête :
cible:[(
  mot~"^[Bb]arrages?$"       | mot~"^BARRAGES?$"       |
  mot~"^[Cc]hefs?$"          | mot~"^CHEFS?$"          |
  mot~"^[Cc]ommunications?$" | mot~"^COMMUNICATIONS?$" |
  mot~"^[Cc]ompagnies?$"     | mot~"^COMPAGNIES?$"     |
  mot~"^[Cc]oncentrations?$" | mot~"^CONCENTRATIONS?$" |
  mot~"^[Cc]onstitutions?$"  | mot~"^CONSTITUTIONS?$"  |
  mot~"^[Dd]egrés?$"         | mot~"^DEGR[EÉ]S?$"      |
  mot~"^[Dd]étentions?$"     | mot~"^D[EÉ]TENTIONS?$"  |
  mot~"^[ÉéE]conomies?$"     | mot~"^[EÉ]CONOMIES?$"   |
  mot~"^[Ff]ormations?$"     | mot~"^FORMATIONS?$"     |
  mot~"^[Ll]ancements?$"     | mot~"^LANCEMENTS?$"     |
  mot~"^[Oo]bservations?$"   | mot~"^OBSERVATIONS?$"   |
  mot~"^[Oo]rganes?$"        | mot~"^ORGANES?$"        |
  mot~"^[Pp]assages?$"       | mot~"^PASSAGES?$"       |
  mot~"^[Pp]ieds?$"          | mot~"^PIEDS?$"          |
  mot~"^[Rr]estaurations?$"  | mot~"^RESTAURATIONS?$"  |
  mot~"^[Ss]olutions?$"      | mot~"^SOLUTIONS?$"      |
  mot~"^[Ss]tations?$"       | mot~"^STATIONS?$"       |
  mot~"^[Ss]uspensions?$"    | mot~"^SUSPENSIONS?$"    |
  mot~"^[Vv]ols?$"           | mot~"^VOLS?$"
) & !(
  lemme="barrage"|lemme="chef"|lemme="communication"|
  lemme="compagnie"|lemme="concentration"|
  lemme="constitution"|lemme="degré"|lemme="détention"|
  lemme="économie"|lemme="formation"|lemme="lancement"|
  lemme="observation"|lemme="organe"|lemme="passage"|
  lemme="pied"|lemme="restauration"|lemme="solution"|
  lemme="station"|lemme="suspension"|lemme="vol"
)]

Apparence :
// Fichier
[P:cible.fichier][C:9]
// Paragraphe
[P:cible.paragraphe][C:9]
// Position dans le paragraphe
[P:cible.position][C:9]
// Contexte gauche
[B:[P:mot][C:32];cible.index-20;cible.index-1][C:9]
// Cible
[P:cible.mot][C:9]
// Lexie
[P:cible.lexie][C:9]
// Contexte droit
[B:[P:mot][C:32];cible.index+1;cible.index+20][C:9]
// Clef contexte gauche
[B:[P:mot][C:32];cible.index-1;cible.index-5][C:9]
// Fin de ligne
[C:10]
```

Figure 4.10 – Règle utilisée pour la détection des noms potentiellement mal lemmatisés.

```
Requête :
cible:[ (ems!~"^N" & (
  mot~"^[Bb]iologiques?$"       | mot~"^BIOLOGIQUES?$"            |
  mot~"^[Cc]laire?s?$"          | mot~"^CLAIRE?S?$"               |
  mot~"^[Cc]orrecte?s?$"        | mot~"^CORRECTE?S?$"             |
  (mot~"^[Cc]ourante?s?$"       | mot~"^COURANTE?S?$")& ems!~"^V" |
  mot~"^[Ee]xceptionnel(le)?s?$"| mot~"^EXCEPTIONNEL(LE)?S?$"     |
  mot~"^[Ff]ra[iî](che)?s?$"    | mot~"^FRA[IÎ](CHE)?S?$"         |
  mot~"^[Hh]aute?s?$"           | mot~"^HAUTE?S?$"                |
  mot~"^[Hh]istoriques?$"       | mot~"^HISTORIQUES?$"            |
  mot~"^[Pp]leine?s?$"          | mot~"^PLEINE?S?$"               |
  mot~"^[Pp]opulaires?$"        | mot~"^POPULAIRES?$"             |
  mot~"^[Rr]éguli(er|ère)s?$"   | mot~"^R[EÉ]GULI(ER|ERE|ÈRE)S?$" |
  mot~"^[Ss]aine?s?$"           | mot~"^SAINE?S?$"                |
  mot~"^[Ss]econdaires?$"       | mot~"^SECONDAIRES?$"            |
  mot~"^[Ss]ensibles?$"         | mot~"^SENSIBLES?$"              |
  mot~"^[Ss]imples?$"           | mot~"^SIMPLES?$"                |
  mot~"^[Ss]tricte?s?$"         | mot~"^STRICTE?S?$"              |
  mot~"^[Ss]ûre?s?$"            | mot~"^S[UÛ]RE?S?$"              |
  mot~"^[Tt]raditionnel(le)?s?$"| mot~"^TRADITIONNEL(LE)?S?$"     |
  mot~"^[Uu]tiles?$"            | mot~"^UTILES?$"                 |
  mot~"^[Vv]astes?$"            | mot~"^VASTES?$"
)) & !(
  lemme="biologique"|lemme="clair"|lemme="correct"|lemme="courant"|
  lemme="exceptionnel"|lemme="frais"|lemme="haut"|lemme="historique"|
  lemme="plein"|lemme="populaire"|lemme="régulier"|lemme="sain"|
  lemme="secondaire"|lemme="sensible"|lemme="simple"|lemme="strict"|
  lemme="sûr"|lemme="traditionnel"|lemme="utile"|lemme="vaste"
)]

Apparence :
// Fichier
[P:cible.fichier][C:9]
// Paragraphe
[P:cible.paragraphe][C:9]
// Position dans le paragraphe
[P:cible.position][C:9]
// Contexte gauche
[B:[P:mot][C:32];cible.index-20;cible.index-1][C:9]
// Cible
[P:cible.mot][C:9]
// Lexie
[P:cible.lexie][C:9]
// Contexte droit
[B:[P:mot][C:32];cible.index+1;cible.index+20][C:9]
// Clef contexte gauche
[B:[P:mot][C:32];cible.index-1;cible.index-5][C:9]
// Fin de ligne
[C:10]
```

Figure 4.11 – Règle utilisée pour la détection des adjectifs potentiellement mal lemmatisés.

mal lemmatisées parmi lesquelles environ 70% le sont vraiment.

La reprise de la procédure d'étiquetage de notre corpus avec une version différente du logiciel CORDIAL ANALYSEUR génère deux types d'incohérences :

- quelques occurrences des 60 vocables que nous n'avions pas encore détectées vont surgir ;
- quelques occurrences des 60 vocables qui avaient bien été détectées vont disparaître.

La première incohérence est due aux modifications et aux améliorations apportées au logiciel CORDIAL ANALYSEUR qui étiquette correctement un jeton qui avait mal été étiqueté par une version précédente et qui n'avait pas été détecté par notre procédure de détection décrite plus haut dans cette section. Pour résoudre ce problème, il suffit de regénérer des fichiers tabulaires (comme décrit dans la section 4.4.1) pour réaliser un étiquetage lexical manuel de ces nouvelles occurrences. La deuxième incohérence entraîne que notre corpus contient des occurrences de nos vocables identifiées et possédant une étiquette lexicale, mais non accessibles par leur lemme (puisque celui-ci est erroné). En fait, ce deuxième type d'incohérence est également généré par la procédure de détection et d'étiquetage lexical manuel des occurrences mal lemmatisées décrite plus haut dans cette section. Ce cas se présente quand une occurrence mal étiquetée par une version antérieure de CORDIAL ANALYSEUR, détectée par notre procédure et corrigée manuellement, reste mal été étiquetée par la nouvelle version de CORDIAL ANALYSEUR. Il faut donc corriger ces lemmes incohérents.

4.4.3 Correction des lemmes incohérents

Nous voulons à tout prix éviter le cas de figure où un jeton possède une étiquette lexicale et où le lemme ne correspond pas à celui de l'un des 60 vocables. Il nous faut éviter cela de manière à pouvoir accéder à toutes les occurrences de nos vocables par leur lemme.

Nous utilisons le logiciel WINLOX pour résoudre ce problème en l'utilisant, dans un premier temps, pour vérifier s'il existe bien des jetons possédant une étiquette lexicale mais dont le lemme ne correspond pas à celui de l'un de nos vocables. L'action du traitement de WINLOX est *Extraire correspondances*. La règle utilisée est présentée figure 4.12. La figure 4.13 montre un extrait du résultat du traitement. Nous identifions ainsi quelques 460 incohérences.

Nous utilisons ensuite WINLOX pour générer le fichier de paramètres qui permet de réaliser un *étiquetage systématique*, toujours avec WINLOX, pour corriger les lemmes. Le fichier généré par un traitement est ici utilisé comme fichier des paramètres d'un nouveau traitement. Avant ce dernier passage de WINLOX, il faut corriger à la main les lemmes erronés dans le fichier des paramètres.

Nous sommes maintenant assurés que :

- la quasi-totalité des occurrences des 60 vocables est correctement lemmatisée ;
- tous les vocables ayant reçu une étiquette lexicale valide sont correctement lemmatisés ;
- tous les mots dont le lemme correspond à celui de l'un des 60 vocables a reçu une étiquette lexicale (valide ou pas suivant que la lemmatisation est correcte ou non comme nous l'avons expliqué dans la section 4.4.1).

```
Requête :
cible:[lexie~"^[0123456789]" &! (
  lemme="entrer"|lemme="mettre"|lemme="venir"|
  lemme="porter"|lemme="passer"|lemme="répondre"|
  lemme="présenter"|lemme="connaître"|lemme="rendre"|
  lemme="comprendre"|lemme="communication"|
  lemme="haut"|lemme="formation"|lemme="ouvrir"|
  lemme="conduire"|lemme="importer"|lemme="chef"|
  lemme="simple"|lemme="tirer"|lemme="arrêter"|
  lemme="poursuivre"|lemme="courant"|lemme="pied"|
  lemme="économie"|lemme="plein"|lemme="solution"|
  lemme="couvrir"|lemme="conclure"|lemme="exercer"|
  lemme="parvenir"|lemme="sûr"|lemme="clair"|
  lemme="historique"|lemme="passage"|lemme="observation"|
  lemme="degré"|lemme="populaire"|lemme="biologique"|
  lemme="frais"|lemme="traditionnel"|lemme="sensible"|
  lemme="constitution"|lemme="compagnie"|lemme="utile"|
  lemme="vaste"|lemme="organe"|lemme="vol"|lemme="station"|
  lemme="concentration"|lemme="exceptionnel"|lemme="strict"|
  lemme="secondaire"|lemme="régulier"|lemme="lancement"|
  lemme="sain"|lemme="correct"|lemme="détention"|
  lemme="suspension"|lemme="restauration"|lemme="barrage"
)]

Apparence :
// Fichier
[P:cible.fichier][C:9]
// Paragraphe
[P:cible.paragraphe][C:9]
// Position dans le paragraphe
[P:cible.position][C:9]
// Jeton de la cible
[P:cible.mot][C:9]
// Lemme de la cible
[P:cible.mot][C:9]
// Lexie
[P:cible.lexie]
// Fin de ligne
[C:10]
```

Figure 4.12 – Règle utilisée pour la détection des jetons possédant une étiquette lexicale mal lemmatisées.

```
O 2733  272 mise      mise      1.2
O 2933  83  importées importé   2.1
O 3510  104 présente  présente  1.2.2
O 4414  53  comprise  compris   1.1.4
O 4463  82  rendue    rendu     1.6
O 4490  33  mise      mise      1.2.1.1
O 4622  78  mises     mise      1.2.1.1
O 4670  54  mise      mise      1.2.1.1
O 4757  120 tiré      tiré      1.4.2
O 4892  39  présente  présent   1.2.2
O 4923  14  compris   compris   1.3
O 5176  14  tirés     tiré      1.4.1
O 5220  81  compris   compris   1.3
O 5251  6   compris   compris   1.3
O 5489  86  hauts     hauts     6
O 5525  15  passée    passé     1.9
O 5525  43  chefs     chefs     4
O 5695  78  exercée   exercé    1.2.1
O 5835  70  porte     porte     1.15
O 6061  48  tiré      tiré      1.4.1
O 6396  160 passée    passé     1.9
O 6554  133 exercé    exercé    1.2.1
```

Figure 4.13 – Extrait du résultat de l'application de la règle 4.12.

4.4.4 Finalisation des corpus

Pour finaliser notre corpus, nous ajoutons une colonne contenant une étiquette morphosyntaxique simplifiée basée sur l'étiquette morphosyntaxique de Cordial Analyseur. La liste de ces étiquettes est présentée en annexe section E.2.

Nous ajoutons également une colonne qui indique, chaque fois que le mot du corpus correspond à l'un des 60 vocables, la fréquence dans le corpus de la lexie de ce vocable. Cette colonne permet, par exemple, de ne s'intéresser qu'aux lexies comportant au moins n occurrences dans le corpus.

Enfin, pour faciliter le travail sur notre corpus, nous effectuons deux opérations supplémentaires. La première est la concaténation des cinq sous-corpus en un seul fichier. Nous pouvons ainsi effectuer tous nos traitements en n'utilisant qu'un fichier unique. Ce fichier occupe sur le disque une taille de 283 311 448 octets, soit environ 270 mégaoctets. La taille de ce fichier est excessivement importante et cause un ralentissement important de nos traitements. Or, la majorité des traitements que nous effectuerons n'implique qu'un vocable à la fois et ne concerne qu'un contexte limité centré sur ce vocable. La deuxième opération effectuée consiste donc en la réalisation de 60 sous-corpus, un par vocable, contenant toutes les occurrences d'un vocable donné, chaque occurrence étant accompagnée d'un contexte de plus ou moins 100 mots (*i.e.* 100 mots avant et 100 mots après). Cette opération est naturellement effectuée avec l'application WinLoX. Ces 60 sous-corpus occupent une taille allant de 40 664 kilo-octets (pour le vocable *mettre*) à seulement 588 kilo-octets (pour le vocable *barrage*). Le plus gros des sous-corpus est 6,8 fois plus petit que le corpus original et le plus petit 471 fois. La durée d'un traitement étant pratiquement proportionnelle à la taille du corpus, sur ces sous-corpus les traitements iront ainsi de 6,8 à 471 fois plus vite suivant le vocable.

La figure 4.14 présente un extrait du corpus finalisé. Sur cette figure, le nom des

fichier	*paragraphe*	*position*	*jeton*	*lemme*	*ems*	*tgb*	*fonc*	*prop*	*smallems*	*lexie*	*freq*
J	215	117	le	le	DETDMS	Da-ms-d	H	1	DET		
J	215	118	maintien	maintien	NCMS	Ncms	H	1	NCOM		
J	215	119	de	de	PREP	Sp	H	1	PREP		
J	215	120	la	le	DETDFS	Da-fs-d	H	1	DET		
J	215	121	sécurité	sécurité	NCFS	Ncfs	H	1	NCOM		
J	215	122	publique	public	ADJFS	Afpfs	H	1	ADJ		
J	215	123	pouvait	pouvoir	VINDI3S	Vmii3s	V	1	VCON		
J	215	124	mettre	mettre	VINF	Vmn–	V	2	VINF	1.12.7	230
J	215	125	fin	fin	NCFS	Ncfs	T	2	NCOM		
J	215	126	à	à	PREP	Sp	F	2	PREP		
J	215	127	la	le	DETDFS	Da-fs-d	F	2	DET		
J	215	128	pratique	pratique	NCFS	Ncfs	F	2	NCOM		
J	215	129	des	de	DETDPIG	Da-.p-i	F	2	DET		
J	215	130	détentions	détention	NCFP	Ncfp	F	2	NCOM	1	81
J	215	131	sans	sans	PREP	Sp	H	2	PREP		
J	215	132	jugement	jugement	NCMS	Ncms	H	2	NCOM		

Figure 4.14 – Extrait du corpus finalisé. Les noms de colonne (*fichier*, *paragraphe*, *position*, *jeton*, *lemme*, *ems*, *tgb*, *fonc*, *prop*, *smallems*, *lexie*, *freq*) correspondent aux noms spécifiés dans le fichier *propriété* pour nommer les colonnes du corpus tabulaire. Ces noms n'apparaissent pas dans le corpus.

colonnes, c'est-à-dire *fichier*, *paragraphe*, *position*, *jeton*, *lemme*, *ems*, *tgb*, *fonc*, *prop*, *smallems*, *lexie* et *freq*, correspondent aux noms spécifiés dans le fichier *propriété* pour nommer les colonnes du corpus tabulaire dans l'application (Win/DOS)LoX. Ce sont ces noms que nous utilisons pour désigner les étiquettes de colonnes particulières dans les méta-expressions régulières et les masques.

4.5 Informations sur les vocables et le corpus

4.5.1 Informations relatives au corpus

Fréquence des catégories grammaticales

Le tableau E.1, en annexe, recense et donne la fréquence des trois types d'étiquettes morphosyntaxiques de tous les mots du corpus.

Taille moyenne d'une phrase en nombre de mots

Le nombre total de mots (ponctuation comprise), donné sur la dernière ligne du tableau E.1, est de 6 468 522. Nous dénombrons 252 104 ponctuations fortes ce qui fait une moyenne de 25,7 mots par phrase.

Taille moyenne d'une phrase en nombre de mots pleins

Nous entendons par mots pleins les noms, les verbes, les adjectifs et les adverbes. Nous dénombrons 1 581 849 noms, 517 688 adjectifs, 669 518 verbes et 304 462 adverbes,

ce qui constitue un total de 1 581 849 + 517 688 + 669 518 + 304 462 = 3 073 517 mots pleins, soit une moyenne de 12,2 mots pleins par phrase.

4.5.2 Liste des formes ambiguës

Un même jeton peut appartenir à plusieurs catégories grammaticales. Par exemple, le jeton *courant* peut être une forme fléchie de :

- l'adjectif *courant* ;
- le nom *courant* ;
- le verbe *courir*.

Le tableau 4.2 donne la liste de toutes les formes fléchies ambiguës des 60 vocables ainsi que les catégories grammaticales possibles de ces formes fléchies.

Jetons	Vocables et catégories grammaticales possibles
CLAIRE	CLAIR (Adj.), CLAIRE (Nom)
CONCLUANT	CONCLUANT (Adj.), CONCLURE (Ver.)
CONDUITE	CONDUIRE (Ver.), CONDUITE (Nom)
CONDUITES	CONDUIRE (Ver.), CONDUITE (Nom)
COURANT	COURANT (Adj.), COURANT (Nom), COURIR (Ver.)
COURANTS	COURANT (Adj.), COURANT (Nom)
COUVERT	COUVERT (Adj.), COUVERT (Nom), COUVRIR (Ver.)
COUVERTE	COUVERT (Adj.), COUVRIR (Ver.)
COUVERTES	COUVERT (Adj.), COUVRIR (Ver.)
COUVERTS	COUVERT (Adj.), COUVERT (Nom), COUVRIR (Ver.)
ENTRAIT	ENTRAIT (Nom), ENTRER (Ver.)
ENTRANT	ENTRANT (Adj.), ENTRANT (Nom), ENTRER (Ver.)
ENTRE	ENTRE (Prép.), ENTRER (Ver.)
ENTRÉE	ENTRER (Ver.), ENTRÉE (Nom)
ENTRÉES	ENTRER (Ver.), ENTRÉE (Nom)
FRAIS	FRAI (Nom), FRAIS (Adj.), FRAIS (Nom), FRAIS (Adv.)
HAUT	HAUT (Adj.), HAUT (Nom), HAUT (Adv.)
HAUTS	HAUT (Adj.), HAUT (Nom)
IMPORTANT	IMPORTANT (Adj.), IMPORTER (Ver.)
METS	METS (Nom), METTRE (Ver.)
MIRENT	METTRE (Ver.), MIRER (Ver.)
MISE	METTRE (Ver.), MISE (Nom), MISER (Ver.)
MISES	METTRE (Ver.), MISE (Nom), MISER (Ver.)
MISSIONS	METTRE (Ver.), MISSION (Nom)
OUVERT	OUVERT (Adj.), OUVRIR (Ver.)
OUVERTE	OUVERT (Adj.), OUVRIR (Ver.)
OUVERTES	OUVERT (Adj.), OUVRIR (Ver.)
OUVERTS	OUVERT (Adj.), OUVRIR (Ver.)
OUVRAIENT	OUVRER (Ver.), OUVRIR (Ver.)
OUVRAIT	OUVRER (Ver.), OUVRIR (Ver.)
OUVRANT	OUVRANT (Adj.), OUVRER (Ver.), OUVRIR (Ver.)
OUVRE	OUVRER (Ver.), OUVRIR (Ver.)
OUVRENT	OUVRER (Ver.), OUVRIR (Ver.)
OUVRES	OUVRER (Ver.), OUVRIR (Ver.)
OUVREZ	OUVRER (Ver.), OUVRIR (Ver.)
OUVRONS	OUVRER (Ver.), OUVRIR (Ver.)
PASSANT	PASSANT (Nom), PASSER (Ver.)
PASSIONS	PASSER (Ver.), PASSION (Nom)

Suite sur la page suivante...

Suite de la page précédente...

Jetons	Vocables et catégories grammaticales possibles
PASSÉ	PASSER (Ver.), PASSÉ (Adj.), PASSÉ (Nom), PASSÉ (Prép.)
PASSÉE	PASSER (Ver.), PASSÉ (Adj.)
PASSÉES	PASSER (Ver.), PASSÉ (Adj.)
PASSÉS	PASSER (Ver.), PASSÉ (Adj.), PASSÉ (Nom)
PLEIN	PLEIN (Adj.), PLEIN (Nom)
PLEINS	PLEIN (Adj.), PLEIN (Nom)
PORTANT	PORTANT (Adj.), PORTER (Ver.)
PORTE	PORTE (Nom), PORTER (Ver.)
PORTER	PORTER (Nom), PORTER (Ver.)
PORTES	PORTE (Nom), PORTER (Ver.)
PORTIONS	PORTER (Ver.), PORTION (Nom)
PORTÉE	PORTER (Ver.), PORTÉE (Nom)
PORTÉES	PORTER (Ver.), PORTÉE (Nom)
POURSUIVANT	POURSUIVANT (Nom), POURSUIVRE (Ver.)
PRÉSENTE	PRÉSENT (Adj.), PRÉSENT (Nom), PRÉSENTER (Ver.)
PRÉSENTES	PRÉSENT (Adj.), PRÉSENT (Nom), PRÉSENTER (Ver.)
VENUE	VENIR (Ver.), VENUE (Nom)
VENUES	VENIR (Ver.), VENUE (Nom)
VIENNE	VENIR (Ver.), VIENNE (Nom)
VINS	VENIR (Ver.), VIN (Nom)

Tableau 4.2 – Liste des formes ambiguës au niveau de la catégorie grammaticale.

4.5.3 Fréquences brutes et après désambiguïsation

Le tableau 4.3 donne la fréquence brute et après désambiguïsation de chacun de nos vocables dans chacun des sous-corpus. Les fréquences brutes sont données dans les colonnes *B*. Les colonnes *D* donnent les fréquences obtenues après désambïguisation. Il ne s'agit pas des fréquences obtenues après une simple désambïguisation basée sur la lemmatisation de Cordial Analyseur, mais des fréquences des occurrences de nos vocables ayant reçu une lexie dans le corpus finalisé.

	ABU		JOC		MON		OUV		PER		**TOTAL**	
Vocables	B	D	B	D	B	D	B	D	B	D	B	D
arrêter	440	425	246	204	184	163	98	59	69	65	1037	916
barrage	1	1	59	59	20	20	8	8	4	4	92	92
biologique	0	0	75	75	15	15	306	306	80	79	476	475
chef	236	236	103	99	483	483	215	215	100	100	1137	1133
clair	266	217	48	44	120	102	99	95	103	99	636	557
communication	20	20	398	397	81	74	80	80	1163	1132	1742	1703
compagnie	70	70	130	130	121	121	67	67	24	24	412	412
comprendre	422	417	429	393	344	321	461	435	599	579	2255	2145
concentration	0	0	108	108	36	36	21	21	84	81	249	246
conclure	55	54	306	306	115	113	136	135	120	119	732	727
conduire	162	144	156	122	230	178	432	288	426	361	1406	1093
connaître	596	511	317	241	422	217	548	391	505	275	2388	1635
constitution	27	26	54	54	108	108	148	148	86	86	423	422
correct	21	20	34	34	11	11	18	18	33	33	117	116
courant	184	6	142	29	127	24	366	58	149	53	968	170
couvrir	234	112	305	286	94	54	53	43	60	48	746	543
degré	129	128	61	61	39	39	105	105	174	174	508	507
détention	1	1	65	65	39	39	6	6	1	1	112	112

Suite sur la page suivante...

Suite de la page précédente...

	ABU		JOC		MON		OUV		PER		**TOTAL**	
Vocables	B	D	B	D	B	D	B	D	B	D	B	D
économie	38	39	218	213	271	275	218	215	190	188	935	930
entrer	1582	555	2135	227	1567	153	2537	182	2510	141	10 331	1258
exceptionnel	6	5	49	49	62	62	64	63	48	47	229	226
exercer	68	66	217	212	93	92	187	183	145	145	710	698
formation	14	14	652	651	203	203	419	419	245	241	1533	1528
frais	139	109	167	55	78	10	73	5	12	5	469	184
haut	631	287	220	180	351	229	174	120	271	201	1647	1017
historique	17	17	83	81	109	107	148	146	273	269	630	620
importer	124	103	409	141	211	76	329	125	333	131	1406	576
lancement	0	0	51	51	22	22	47	47	18	18	138	138
mettre	1322	1268	2349	1193	1124	816	1377	899	1640	1070	7812	5246
observation	122	122	77	77	28	28	124	124	221	221	572	572
organe	31	31	77	77	29	29	199	199	30	30	366	366
ouvrir	538	367	217	110	319	171	160	95	295	176	1529	919
parvenir	129	118	147	146	131	130	154	149	112	111	673	654
passage	107	107	48	48	86	86	175	175	185	184	601	600
passer	1110	987	226	142	860	632	611	395	656	400	3463	2556
pied	690	689	59	59	157	151	32	32	29	29	967	960
plein	488	473	78	76	163	143	95	90	65	62	889	844
populaire	62	18	42	42	142	141	87	87	175	169	508	457
porter	1249	566	639	559	633	368	610	444	612	410	3743	2347
poursuivre	121	121	271	271	169	169	324	324	93	93	978	978
présenter	192	181	853	774	334	289	485	385	638	513	2502	2142
régulier	35	35	63	63	24	24	28	28	31	31	181	181
rendre	458	395	337	293	451	366	510	458	530	478	2286	1990
répondre	1413	1413	401	397	232	232	249	248	246	239	2541	2529
restauration	2	2	49	49	25	25	19	19	9	9	104	104
sain	16	16	25	25	20	20	18	18	51	50	130	129
secondaire	9	9	41	41	43	38	66	65	44	42	203	195
sensible	53	51	95	95	66	66	114	111	108	102	436	425
simple	233	223	55	54	171	165	316	314	298	295	1073	1051
solution	14	14	200	199	145	145	351	351	176	171	886	880
station	10	7	79	79	79	79	35	35	67	66	270	266
strict	5	5	80	80	32	32	61	61	42	42	220	220
sûr	148	148	48	48	213	213	97	97	139	139	645	645
suspension	0	0	58	58	17	17	17	17	18	18	110	110
tirer	464	445	97	93	173	162	169	159	152	143	1055	1002
traditionnel	0	0	81	81	105	105	134	134	127	127	447	447
utile	55	52	102	102	39	39	101	98	72	68	369	359
vaste	121	121	58	58	55	55	70	70	64	64	368	368
venir	1903	1798	322	242	934	823	441	405	553	529	4153	3797
vol	63	63	94	94	35	35	21	21	65	65	278	278
TOTAL	16646	13428	14605	10292	12590	9141	14613	10090	15368	10845	73822	53796

Tableau 4.3 – Fréquences brutes (colonne B) et après désambiguïsation (colonne D) des 60 vocables de l'étude.

4.5.4 Répartition des lexies par vocable

En annexe D, le tableau D.1 précise pour chacune des lexies de chacun de nos vocables sa fréquence dans le corpus et dans chacun des sous-corpus.

Les tableaux 4.4, 4.5 et 4.6 montrent l'ensemble des 60 vocables de notre étude répartis en 20 noms, 20 adjectifs et 20 verbes. Pour chacun de ces vocables, ces tableaux donnent une information sur sa fréquence (colonne *fréquence*), sur le nombre de ses lexies (colonne *lexie*), sur l'entropie de la répartition des occurrences sur ses lexies (colonne H) et enfin, à entropie égale, sur le nombre de lexies qu'aurait ce vocable si ses lexies étaient équiprobables (colonne perplexité : 2^H).

La grande disparité de la fréquence de ces vocables est notable. Le moins fréquent est *barrage*, avec une fréquence de 92, et le plus fréquent est le verbe *mettre*, avec une fréquence de 5246. Il faut également remarquer que le nombre de lexies peut être très important (62 pour *pied*, 35 pour *plein*, 84 pour *passer*, 140 pour *mettre*, etc.). Comme nous l'avons précisé dans la section 4.2.2, nos lexies correspondent plus à des usages qu'à des sens. Nous avons choisi d'affecter une lexie différente pour chaque emploi du vocable dans une construction figée ou semi-figée. Voilà pourquoi certains vocables comportent un grand nombre de lexies. Le vocable *pied*, dont l'entrée du dictionnaire est présentée en annexe C.5, illustre bien ce phénomène puisqu'il intervient dans un grand nombre de constructions figées ou semi-figées (comme : *à* ***pied*** *d'œuvre*, *de* ***pied*** *en cap*, *être* ***pieds*** *et poings liés*, ***pied*** *de nez*, ***pied*** *de page*, etc.). Nous travaillons donc avec une granularité au niveau des sens relativement importante : plus de 14 lexies par vocable (en réalisant une moyenne pondérée) pour les noms et les adjectifs, et plus de 47 pour les verbes.

Cependant, le nombre de lexies n'est pas un bon indice de la difficulté de la tâche. En effet, il est plus facile de lever l'ambiguïté d'un vocable ayant 10 lexies, mais dont la quasi-totalité des occurrences est regroupée sous une seule lexie, que de lever l'ambiguïté d'un vocable comportant deux lexies équiprobables. L'entropie de la répartition des occurrences du vocable sur ses différentes lexies est un meilleur indicateur de la difficulté de la levée de l'ambiguïté pour ce vocable, d'où la présence de la colonne H, pour l'entropie, et de la colonne 2^H qui mesure la perplexité, ce qui peut s'avérer plus parlant. La perplexité indique le nombre de lexies équiprobables qu'il faudrait pour obtenir une entropie équivalente. Ainsi, pour une entropie inchangée, si les lexies étaient équiprobables, les noms auraient une moyenne (pondérée) de 4,5 lexies par vocables, les adjectifs une moyenne de 6,3 et les verbes de 9,9. Il s'agit là d'un indice qui laisse présager une plus grande difficulté pour lever l'ambiguïté des adjectifs, et encore plus des verbes, par rapport aux noms.

Vocables	*fréquence*	*lexie*	H	2^H
barrage	92	5	1,2	2,3
chef	1133	11	1,5	2,8
communication	1703	13	2,4	5,4
compagnie	412	12	1,6	3,1
concentration	246	6	2	3,9
constitution	422	6	1,6	3,1
degré	507	18	2,5	5,5
détention	112	2	0,9	1,8
économie	930	10	2,2	4,5
formation	1528	9	1,7	3,2
lancement	138	5	1	2
observation	572	3	0,7	1,6
organe	366	6	2,2	4,7
passage	600	19	2,7	6,5
pied	960	62	3,5	11,7
restauration	104	5	1,8	3,6
solution	880	4	0,4	1,4
station	266	8	2,6	6
suspension	110	5	1,5	2,8
vol	278	10	2,2	4,6
Moyenne pondérée	**568**	**14,2**	**1,9**	**4,5**
Fréquence des occurrences : 11 359				

Tableau 4.4 – Informations sur les 20 noms de l'étude. Pour chaque vocable, ce tableau précise la fréquence (colonne *fréquence*), le nombre de lexies (colonne *lexie*), l'entropie de la répartition des occurrences sur les lexies (colonne H) et enfin, à entropie égale, le nombre de lexies qu'aurait le vocable si ses lexies étaient équiprobables (colonne perplexité : 2^H).

Vocables	*fréquence*	*lexie*	H	2^H
biologique	475	4	0,5	1,5
clair	557	20	3,1	8,6
correct	116	5	1,8	3,5
courant	170	4	0,6	1,5
exceptionnel	226	3	1,4	2,7
frais	184	18	3,1	8,7
haut	1017	29	3,5	11
historique	620	3	0,7	1,6
plein	844	35	4	15,9
populaire	457	5	2	4
régulier	181	11	2,5	5,8
sain	129	10	2,4	5,5
secondaire	195	5	1,7	3,2
sensible	425	11	2,6	6,2
simple	1051	14	2,1	4,4
strict	220	9	2,2	4,7
sûr	645	14	2,6	6,1
traditionnel	447	2	0,5	1,4
utile	359	9	2,4	5,2
vaste	368	6	2,1	4,2
Moyenne pondérée	**434,3**	**14,1**	**2,3**	**6,3**
Fréquence des occurrences : 8686				

Tableau 4.5 – Informations sur les 20 adjectifs de l'étude. La description de chacune des colonnes est donnée dans la légende du tableau 4.4.

Vocables	*fréquence*	*lexie*	H	2^H
arrêter	916	15	3	7,8
comprendre	2145	13	2,8	6,8
conclure	727	16	2,4	5,1
conduire	1093	15	2,3	4,8
connaître	1635	16	2,2	4,7
couvrir	543	22	3,3	9,7
entrer	1258	39	3,7	12,8
exercer	698	8	1,5	2,9
importer	576	8	2,6	5,9
mettre	5246	140	3,7	12,7
ouvrir	919	41	3,8	13,9
parvenir	654	8	2,3	5
passer	2556	84	4,5	22,5
porter	2347	59	4	16,3
poursuivre	978	16	2,7	6,5
présenter	2142	18	2,6	5,9
rendre	1990	27	2,9	7,4
répondre	2529	9	1	2
tirer	1002	47	3,9	14,8
venir	3797	33	3,2	9,3
Moyenne pondérée	**1687,6**	**47,4**	**3,1**	**9,9**
Fréquence des occurrences : 33 751				

Tableau 4.6 – Informations sur les 20 verbes de l'étude. La description de chacune des colonnes est donnée dans la légende du tableau 4.4.

Chapitre 5

Méthodologie et étude préliminaire

5.1 Introduction

5.1.1 Problématique

Nous disposons d'un corpus où les occurrences de 60 vocables sont lexicalement désambiguïsées. Nous désirons concevoir un outil de désambïguisation lexicale capable de désambiguïser automatiquement de nouvelles instances de ces vocables dans de nouvelles phrases. Ce type de problème entre dans la catégorie des problèmes auxquels les techniques d'apprentissage supervisé s'adressent.

En IA, les premiers travaux sur l'apprentissage remontent probablement aux travaux sur les perceptrons de Rosenblatt. Une façon simple d'apprendre est d'observer des situations pour lesquelles la réponse adéquate est connue. Un système apprend s'il profite de cette expérience pour construire une base de connaissances qui lui permet de prendre ensuite une décision face à une situation dans laquelle la réponse est inconnue. Ce cadre très général constitue le fondement de l'apprentissage supervisé.

Les techniques d'apprentissage supervisé sont utilisées par de nombreuses équipes de recherche pour tenter de solutionner notre problème de désambïguisation lexicale (cf. section 2.5.2). Nous avons donc choisi d'utiliser ces techniques pour concevoir notre outil de désambïguisation lexicale automatique. Cet outil doit passer par une phase d'apprentissage sur une partie de notre corpus. À l'issue de cette phase, cet outil doit être capable de déterminer automatiquement la bonne lexie de chacune des occurrences des 60 vocables dans de nouveaux corpus. Pour évaluer ses performances, nous utilisons la partie de corpus mise de côté lors de la phase d'apprentissage.

La conception d'un tel outil de désambiguïsation lexicale nécessite :

- des compétences en linguistique et en linguistique informatique pour l'élaboration des critères ;
- des compétences en algorithmique et en mathématique pour la conception des algorithmes d'apprentissage.

Plutôt que de concevoir un outil qui combine ces deux facettes à l'intérieur d'un même module, nous avons préféré les dissocier. Ainsi, notre outil de désambiguïsation est constitué de deux modules. Le premier module permet la mise en œuvre de un ou plusieurs critères susceptibles d'être pertinents pour la levée de l'ambiguïté lexicale.

La tâche de ce module est de produire les données sur lesquelles le second module opérera. Ce second module met en œuvre des techniques d'apprentissage supervisé. Cette approche modulaire nous permet de travailler sur chacun des modules de manière indépendante : nous pouvons changer à volonté d'algorithme de classification et nous pouvons changer ou combiner à volonté les critères de désambiguïsation.

Le premier module est déjà réalisé et est présenté en détail en annexe A. En effet, l'application (WIN/DOS)LOX permet de modéliser des critères, de les appliquer au corpus et de générer les données pour le second module.

5.1.2 Interaction algorithme de classification/critère

Les performances de cet outil de désambiguïsation lexicale sont fonction de la pertinence du ou des critères choisis, mais dépendent également des capacités de l'algorithme de classification. Pour un critère donné, l'estimation de la qualité de la désambiguïsation obtenue avec différents algorithmes de classification nous donne une idée de la performance relative de ces algorithmes. D'autre part, pour un algorithme de classification donné, l'estimation de la qualité de la désambiguïsation obtenue avec différents critères de désambiguïsation nous donne une idée sur le pouvoir discriminant relatif de ces critères.

L'objet de ce travail de thèse n'est pas de mener de front une étude approfondie sur les critères de désambiguïsation lexicale automatique et une autre sur les algorithmes de classification. Notre effort porte principalement sur les critères de désambiguïsation lexicale automatique. Toutefois, négliger totalement le second aspect ne serait certainement pas judicieux. En effet, il existe un grand nombre d'algorithmes de classification et leurs performances sont inégales et dépendantes du domaine. D'autre part, il faut forcément ajuster les paramètres de l'algorithme de classification choisi au problème donné (Daelemans & Hoste, 2002). Enfin, il existe plusieurs types d'algorithmes de classification (algorithmes de type bayesien, de type liste de décisions, de type arbre de décision, de type basé instance, de type réseau de neurones, etc.). Des algorithmes de types différents peuvent réagir de manière contradictoire à deux critères distincts (*i.e.* le critère donnant les meilleurs résultats pour un algorithme donné n'est pas forcément celui qui donne les meilleurs résultats avec un autre algorithme de classification) ou à la combinaison de plusieurs critères. Il peut donc s'avérer judicieux d'utiliser conjointement différents types d'algorithmes pour juger de la pertinence d'un critère. Aussi nous paraît-il important d'évaluer et d'ajuster plusieurs algorithmes de classification afin de choisir celui, ou ceux, que nous utiliserons pour comparer les performances des critères de désambiguïsation.

Nous nous trouvons maintenant face à un dilemme :

- pour tester et ajuster un algorithme de classification, nous avons besoin d'un critère de désambiguïsation lexicale fiable, c'est-à-dire un critère qui génère des indices pertinents pour la levée de l'ambiguïté lexicale ;
- d'un autre côté, pour évaluer et affiner des critères de désambiguïsation lexicale, nous avons besoin d'un ou de plusieurs algorithmes de classification également fiables.

Pour résoudre ce dilemme, nous allons réaliser une étude préliminaire sur un critère reconnu comme fiable par d'autres équipes, celui des cooccurrences [1], en utilisant un

1. Kelly et Stone (1975), Yarowsky (1993), Mooney (1996), Ng et Lee (1996), Agirre et Martinez (2001a), entre autres, montrent que les cooccurrences constituent un bon critère pour identifier le sens d'un mot.

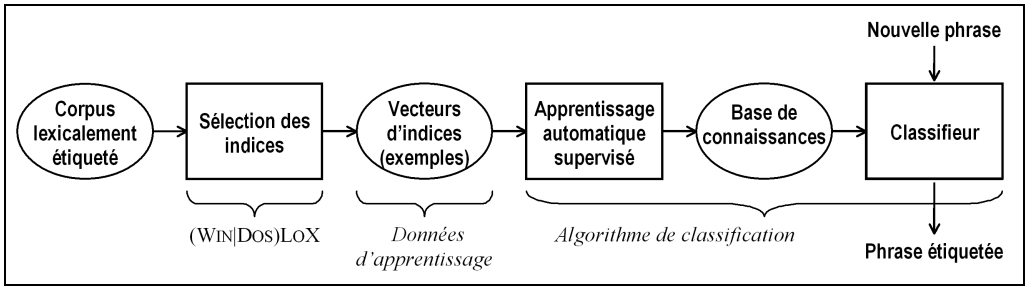


Figure 5.1 – Schéma de principe de la désambiguïsation lexicale supervisée.

algorithme de classification simple à mettre en œuvre et existant du type *liste de décisions*. Nous allons utiliser cet algorithme de classification pour évaluer et affiner notre critère basé sur les cooccurrences. L'objectif n'est absolument pas d'obtenir un critère optimal, mais plutôt de définir un critère relativement fiable. Nous pourrons alors, en utilisant ce critère, établir et ajuster notre palette d'algorithmes de classification (objet du chapitre 6). Tous les outils seront alors en place pour mener à bien notre étude des critères pour la désambiguïsation lexicale automatique (objet du chapitre 7).

5.1.3 Présentation des sections

Dans ce chapitre, nous décrivons tout d'abord le principe de la désambiguïsation lexicale supervisée, puis les étapes de la mise en œuvre d'un critère (**section 5.2**). Nous présentons également dans cette section l'application d'un algorithme de classification, ainsi que les différentes mesures de la qualité de la classification réalisée.

Nous effectuons ensuite une étude préliminaire (**section 5.3**) dont les objectifs sont d'une part, de valider notre approche et nos outils, et d'autre part, d'énoncer un premier critère sur lequel nous pourrons nous appuyer pour mettre au point nos différents algorithmes de classification.

5.2 Méthodologie

5.2.1 Principe de la désambiguïsation lexicale supervisée

Le schéma de la figure 5.1 illustre le principe général de la désambiguïsation lexicale supervisée. Comme le montre ce graphique, le point de départ est un corpus lexicalement étiqueté (dans notre cas, le corpus qui a fait l'objet du chapitre précédent). Sur ce corpus, nous devons sélectionner des indices pour générer une liste de vecteurs. Comme nous le verrons dans la section 6.2.1, nous appelons ces vecteurs des exemples. Lorsque nous travaillons sur un vocable, chaque occurrence de ce vocable dans le corpus doit générer un exemple. Un exemple contient une liste d'indices et la lexie du mot qu'il décrit. Ces données sont générées par l'application d'un critère sur le corpus. C'est donc l'application (WIN/DOS)LOX qui réalise le traitement correspondant au module « Sélection des indices » du graphique.

La section 5.2.2 décrit ce qu'est un critère et comment nous le formalisons. La section 5.2.3 décrit l'application d'un critère sur le corpus pour générer la liste d'exemples qui constitue les données d'apprentissage. L'étape suivante consiste à mettre en œuvre des techniques d'apprentissage supervisé pour générer la base de connaissances qui permet au classifieur de réaliser un étiquetage lexical automatique sur de nouvelles

phrases. Cette étape est très sommairement décrite dans la section 5.2.4 et fait l'objet du chapitre 6. Dans la section 5.2.5, nous exposons comment nous mesurons les performances de l'étiquetage lexical automatique, c'est-à-dire les performances du module « Classifieur » sur le graphique. Les modules « Apprentissage automatique supervisé » et « Classifieur » sont intimement liés et parfois désignés par le terme d'*algorithme de classification*. La mise en œuvre d'un algorithme de classification comporte deux phases correspondant aux deux modules du graphique : une phase d'apprentissage et une phase d'exploitation.

5.2.2 Énonciation d'un critère

La première étape de l'évaluation de la pertinence d'un critère, dans une perspective de désambiguïsation lexicale automatique, est l'énonciation de ce critère. Voici, tout d'abord, une définition de ce que nous entendons par critère :

Définition 5.1 – Critère –
Un critère désigne un ensemble de phénomènes linguistiques susceptibles de survenir dans le contexte d'un vocable.

Il n'est pas possible d'utiliser toute l'information du contexte du mot à désambiguïser car celle-ci est trop bruitée. Il faut donc se focaliser sur un certain nombre d'indices. Le rôle d'un critère est justement de fournir un ensemble d'indices susceptibles d'être pertinents pour la levée de l'ambiguïté.

Voici un exemple d'énoncé de critère :

> *le lemme des trois noms, adjectifs ou verbes qui précèdent ou qui suivent le vocable étudié dans la phrase.*

En étudiant le vocable *détention*, ce critère désigne dans la phrase

> « La Commission a demandé à la Délégation à Yaoundé d'intervenir auprès des autorités camerounaises en *vue* d'une *amélioration* des *conditions* de ***détention*** à la *prison* de *Tcholliré*. »

la liste des lemmes suivants : *vu*, *amélioration*, *condition*, *prison*, *Tcholliré*.

La première étape de la mise en œuvre d'un critère consiste à trouver, puis à énoncer clairement un critère susceptible d'être pertinent pour la levée de l'ambiguïté lexicale. De plus, ce critère doit pouvoir être mis en œuvre dans une perspective de traitement automatique. Il est, par exemple, impossible d'utiliser directement tous les critères différentiels destinés à un étiquetage lexical manuel, comme ceux présentés par Reymond (2001). En effet, les critères différentiels sont souvent difficiles, voire impossibles, à mettre en œuvre car ils font intervenir le jugement du linguiste. De plus, les critères qui font intervenir des classes d'objets, ou des informations syntaxiques, imposent de disposer de ces informations. Or, si la lemmatisation ou l'étiquetage morphosyntaxique sont des techniques qui commencent à arriver à maturité, l'appartenance à une classe d'objets (Gross & Clas, 1997) et l'étiquetage syntaxique sont encore du domaine de la recherche. À l'inverse, la puissance de calcul des ordinateurs permet d'utiliser des critères difficiles à mettre en œuvre manuellement.

La deuxième étape de la mise en œuvre d'un critère est sa modélisation. La modélisation du critère énoncé consiste en l'écriture d'une règle constituée d'une requête et d'un masque. La requête désigne le ou les phénomènes linguistiques auxquels le critère s'intéresse. Le masque permet de spécifier comment sont formatés les résultats de la requête. Nous rappelons que les requêtes sont basées sur des méta-expressions régulières (*i.e.* qui se situent au niveau des mots) et qu'elles permettent de décrire de manière

formelle des classes de suites de mots, donc des portions de texte. La syntaxe exacte des règles, des requêtes, et des masques est détaillée section 3.4.

Supposons, par exemple, un critère dont l'énoncé est :

> *Lemme des trois noms, adjectifs, verbes ou adverbes qui suivent le mot détention sans sortir de la phrase.*

Une modélisation possible de l'énoncé de ce critère peut être réalisée par la requête :

```
[lemme="détention"]
([ems!~"(^N|^ADV|^V|^ADJ)"]*
 [ems~"(^N|^ADV|^V|^ADJ)"]  ){1,3}
stop(ems="PCTFORTE")
```

utilisée conjointement avec le masque :

```
[P:end.lemme]
```

La requête retourne toutes les correspondances possibles qui :

- commencent par une occurrence dont le lemme est *détention* ;
- se terminent par un nom, un adjectif, un verbe ou un adverbe ;
- contiennent un à trois noms, adjectifs, verbes ou adverbes ;
- ne contiennent pas de ponctuation forte.

Le masque `[P:end.lemme]` permet de retourner le lemme du dernier mot de la correspondance à laquelle il est appliqué.

Pour illustrer notre propos, prenons l'extrait de corpus ci-dessous :

> « Le juge , Jean - Marie Lion , a sanctionné cette erreur de trajectoire d ' une inculpation d ' homicide volontaire . Le boulanger a été inculpé de **détention** illégale d ' armes de quatrième catégorie . Cinq amis d ' Ali Rafa ont été inculpés de vols , dégradation volontaire et voies de fait pour . . . »

Appliquée à cet extrait de corpus, notre requête retourne trois correspondances qui désignent les portions de corpus suivante :

1. *détention illégale* ;
2. *détention illégale d ' armes* ;
3. *détention illégale d ' armes de quatrième.*

Le masque va formater ces correspondances de la manière suivante :

1. *illégal* ;
2. *arme* ;
3. *quatrième.*

Nous verrons dans la section suivante l'écriture complète de la règle correspondant à ce critère.

5.2.3 Application d'un critère au corpus

L'application du critère au corpus doit produire un fichier contenant toutes les informations utiles à l'apprentissage et à l'évaluation des algorithmes de classification. Comme nous le verrons dans la section 6, pour l'apprentissage et pour l'évaluation des algorithmes de classification, nous avons besoin, pour chaque exemple, de sa description et de sa classification. Ainsi, le format du fichier résultant de l'application du critère au corpus, doit comporter autant de lignes que d'occurrences du vocable étudié dans le corpus. Chaque ligne correspond à un exemple et contient les informations suivantes :

- un identifiant de l'exemple, c'est-à-dire la référence de l'occurrence dans le corpus ;
- la classe de l'exemple, c'est-à-dire la lexie de cette occurrence ;

– la description qui consiste en l'énumération de tous les phénomènes linguistiques désignés par le critère.

Lorsque nous utilisons l'action *Dénombrer les correspondances* de l'application (Win/DOS)LoX, le manuel précise (cf. section A.4.6 de l'annexe A) :

> *Dans le cas où le discriminant est utilisé, le fichier référence comporte autant de lignes que de couples* { chaîne générée par le masque apparence / chaîne générée par le masque discriminant } *différents générés. Chacune des lignes contient la chaîne de caractères générée par le masque apparence suivie de la chaîne de caractères générée par le masque discriminant suivie de la liste des chaînes de caractères générées par le masque référence. Chacune de ces chaînes est séparée par un séparateur de colonne. Il peut très bien y avoir plusieurs références identiques sur une même ligne.*

C'est exactement ce dont nous avons besoin. Pour générer le fichier que nous désirons, nous utilisons l'action *Dénombrer les correspondances* de manière singulière puisque nous n'avons pas besoin d'un dénombrement. Pour cette raison, nous n'utilisons pas le fichier *Résultats* mais seulement le fichier *Références*.

La règle complète, correspondant à l'énoncé du critère de la section précédente, est :

```
Requête :
cible:[lemme="détention"]
([ems!~"(^N|^ADV|^V|^ADJ)"]*[ems~"(^N|^ADV|^V|^ADJ)"]){1,3}
stop(ems="PCTFORTE")

Apparence :
[P:cible.fichier]_[P:cible.paragraphe]_[P:cible.position]

Référence : [P:end.lemme]

Discriminant : [P:cible.lexie]
```

Figure 5.2 – Règle permettant de modéliser le critère « Lemme des trois noms, adjectifs, verbes ou adverbes qui suivent le mot détention sans sortir de la phrase ».

L'application de ce critère sur notre corpus produit un fichier de la forme suivante :

Identifiant (Apparence)	*Lexie (Discriminant)*	*Énumération des phénomènes linguistiques (Référence)*		
J_10099_3	1	coopération	demander	envisager
J_10130_3	1	affaire	ministre	réunir
J_10237_278	1	camp	charger	rapport
J_11464_36	2	accise	circulation	contrôle
J_20035_48	1	communautaire	diminution	durée
J_20892_14	2	acquisition	arme	concerner
J_215_130	1	communauté	estimer	fin

Figure 5.3 – Extrait d'un fichier contenant des exemples d'apprentissage généré par l'application de la règle de la figure 5.2. Les chaînes de caractères de la première colonne sont générées par le masque *Apparence*, celles de la seconde par le masque *Discriminant* et celles de la dernière par le masque *Référence*.

5.2.4 Application d'un algorithme de classification

Le fichier précédent (figure 5.3) constitue notre échantillon d'exemples. Pour l'exploiter, il faut mettre en œuvre des techniques d'apprentissage à partir d'exemples pour inférer une procédure de classification automatique. Nous détaillons ces techniques dans le chapitre 6. Pour le moment, disons simplement que, grâce à une méthode appelée *validation croisée k-fois* (cf. section 6.2.5), la procédure de classification automatique inférée est utilisée pour réaliser une classification sur l'ensemble des occurrences de notre corpus. La procédure de classification cherchant à affecter une lexie à chacune des occurrences du vocable étudié, deux cas de figure se présentent :

- la procédure de classification a affecté la bonne lexie ;
- la procédure de classification a affecté la mauvaise lexie ;

L'application d'un algorithme de classification sur le fichier d'exemples donne, pour le vocable étudié, la fréquence de ces deux cas de figure ainsi que le nombre d'instances rencontrées. À l'issue de l'application d'un algorithme de classification, nous obtenons donc les informations suivantes :

- nombre de classifications correctes ;
- nombre de classifications effectuées ;
- nombre d'instances rencontrées.

Ces trois informations nous permettent de mesurer la qualité de la classification.

5.2.5 Mesures de la qualité de la classification

Nous allons ici donner la définition de certaines mesures qui nous permettront de juger de la qualité de la classification. La plus classique de ces mesures est la précision.

Mesure de la précision

Définition 5.2 – Précision d'un algorithme –
Nous appellerons précision d'un algorithme le rapport entre le nombre de classifications correctes et le nombre de classifications effectuées par cet algorithme :

$$Précision(Algorithme) = \frac{Nombre\ de\ classifications\ correctes}{Nombre\ de\ classifications\ effectuées} .$$

Tout algorithme doit être comparé, en terme de performance, à l'algorithme majoritaire qui associe à toute description la classe la plus fréquente.

Définition 5.3 – Algorithme majoritaire –
L'algorithme majoritaire associe à toute description la classe la plus fréquente.

Une procédure de classification induite doit toujours au moins dépasser le pouvoir prédictif de l'*algorithme majoritaire.*

Définition 5.4 – Précision de l'algorithme majoritaire –
Nous appellerons précision de l'algorithme majoritaire («base line» en anglais) la précision obtenue en utilisant l'algorithme majoritaire.

Mesure de l'amélioration par rapport à la précision de l'algorithme majoritaire

La mesure de précision ne donne pas une indication satisfaisante. En effet, mieux vaut un algorithme donnant une précision de 60% alors que la précision de l'algorithme

majoritaire est de 30%, ce qui constitue une bonne amélioration, qu'un algorithme donnant une précision de 80% alors que la précision de l'algorithme majoritaire est de 90%, ce qui correspond à une nette dégradation. Aussi parlerons-nous souvent de gain par rapport à l'algorithme majoritaire (ou, autrement dit, de réduction de l'erreur par rapport à l'algorithme majoritaire).

Définition 5.5 – Gain d'un algorithme –
Nous appellerons gain d'un algorithme de classification l'amélioration de la précision obtenue par cet algorithme par rapport à la précision de l'algorithme majoritaire :

$$Gain(Algorithme) = \frac{Précision(Algorithme) - Précision(Algorithme\,majoritaire)}{1 - Précision(Algorithme\ majoritaire)} \ .$$

Mesure du rappel

Cependant, le gain n'est pas tout. En effet, quel est l'intérêt d'un algorithme dont le gain est de 99% mais qui ne classe que 1% des exemples ? C'est pourquoi nous introduisons une nouvelle mesure, le rappel, qui permet de mesurer la couverture d'un algorithme.

Définition 5.6 – Rappel d'un algorithme –
Le rappel d'un algorithme de classification mesure le rapport entre le nombre de classifications correctes effectuées par l'algorithme et le nombre d'exemples à classer :

$$Rappel(Algorithme) = \frac{Nombre\ de\ classifications\ correctes}{Nombre\ d'exemples\ à\ classer} \ .$$

Il faut noter ici une petite particularité par rapport à des disciplines comme la recherche de document où le rappel est défini de la manière suivante :

$$Rappel(Algorithme) = \frac{\text{Nombre de documents pertinents trouvés}}{\text{Nombre total de documents pertinents}} \ .$$

Il est alors aisé d'obtenir un rappel de 100% en retournant tous les documents, au détriment de la précision qui est alors très faible. Il en est tout autrement dans le cadre de notre étude. En effet, nous ne travaillons que sur des algorithmes de classification qui prennent une décision unique (correcte ou non) ou n'en prennent pas. Dans tous les cas, le nombre de classifications effectuées est forcement inférieur ou égal au nombre d'instances à classer, or :

$$Rappel = \frac{\text{Nb. classifications correctes}}{\text{Nb. exemples à classer}} \text{ et } Précision = \frac{\text{Nb. classifications correctes}}{\text{Nb. classifications effectuées}}$$

donc la précision d'un algorithme est toujours supérieure ou égale à son rappel dans notre étude ($Précision(Algorithme) \geq Rappel(Algorithme)$). Il y a égalité quand l'algorithme prend une décision sur toutes les occurrences.

Mesure combinée du rappel et du gain

Il est difficile de comparer les performances de deux algorithmes qui ne prennent pas de décision sur toutes les occurrences et qui ont donc un gain et une précision différente. Pour comparer de tels algorithmes, nous ajoutons une mesure qui combine le gain et le rappel. Quand l'un des deux s'approche de zéro cette mesure doit aussi s'approcher de zéro. Nous désirons également que cette mesure soit identique pour un algorithme

ayant un gain de 80% et un rappel de 40% et un algorithme ayant un gain de 40% et un rappel de 80%. La *F-mesure* (Rijsbergen, 1979) combine le rappel et la précision en une unique mesure d'efficacité.

Définition 5.7 – F-mesure –
La F-mesure *permet de combiner en une seule valeur les mesures de précision (P) et de rappel (R) :*

$$F\text{-}mesure = \frac{(\beta^2+1).P.R}{\beta^2.P+R} \ .$$

Le paramètre β permet de régler les influences respectives de la précision P et du rappel R. Il est très souvent fixé à 1.

En remplaçant la précision par le gain, et en prenant $\beta = 1$, cette mesure répond tout à fait à notre cahier des charges.

Définition 5.8 – Performance d'un algorithme –
Nous appellerons performance d'un algorithme la moyenne harmonique de son rappel et de son gain :

$$Performance(Algorithme) = \frac{2.Rappel(Algorithme).Gain(Algorithme)}{Rappel(Algorithme)+Gain(Algorithme)} \ .$$

Cette formule est utilisable si $Gain(Algorithme) \geq 0$. Quand $Gain(Algorithme) < 0$ nous posons $Performance(Algorithme) = 0$.

Classification de toutes les instances

Quant un algorithme effectue une classification de toutes les instances à classer, nous avons forcément :

$$Précision(Algorithme) = Rappel(Algorithme) \ .$$

Il est facile de se ramener à un algorithme qui effectue une classification sur toutes les instances à classer en affectant à toutes les instances non classées la classe majoritaire.

Mesure moyenne

Nous aurons régulièrement à effectuer une moyenne, sur plusieurs vocables, de la précision ou du rappel. En effet, nous parlerons régulièrement, par exemple, de la précision moyenne sur les 20 noms, les 20 adjectifs, les 20 verbes ou même sur l'ensemble des 60 vocables. Pour effectuer de telles moyennes, nous n'effectuons pas des moyennes arithmétiques simples mais des moyennes arithmétiques pondérées pour tenir compte du fait que les précisions et rappels sont calculés sur des effectifs différents.

Soit, par exemple, respectivement, C_A et C_B les nombres de classifications correctes et E_A et E_B les nombres de classifications effectuées par l'algorithme sur deux ensembles d'exemples A et B de cardinal respectifs $Card(A)$ et $Card(B)$. Les précisions et rappels de cet algorithme sur ces deux ensembles d'exemples sont :

$$P_A = \frac{C_A}{E_A} \quad ; \quad P_B = \frac{C_B}{E_B} \quad ; \quad R_A = \frac{C_A}{Card(A)} \quad ; \quad R_B = \frac{C_B}{Card(B)} \ .$$

La précision et le rappel moyen de cet algorithme sur ces deux ensembles d'exemples est :

$$Pm = \frac{E_A.P_A + E_B.P_B}{E_A+E_B} \qquad \text{et} \qquad Rm = \frac{Card(A).R_A + Card(B).R_B}{Card(A)+Card(B)}$$

et non pas :

$$Pm = \frac{P_A + P_B}{2} \quad \text{et} \quad Rm = \frac{R_A + R_B}{2} \ .$$

Une autre façon de calculer la précision et le rappel moyen est :

$$Pm = \frac{C_A + C_B}{E_A + E_B} \qquad \text{et} \qquad Rm = \frac{C_A + C_B}{Card(A) + Card(B)}$$

ce qui revient, d'une certaine façon, à calculer la précision et le rappel sur la réunion des deux ensembles d'exemples.

L'inconvénient majeur de calculer une moyenne arithmétique pondérée est que nous privilégions ainsi les vocables les plus fréquents au détriment des vocables peu fréquents. Alors pourquoi effectuer une moyenne arithmétique pondérée plutôt qu'une moyenne arithmétique simple ? Pour que le calcul de la moyenne sur des éléments dont les effectifs ne sont pas égaux soit cohérent, il faut effectuer une moyenne arithmétique pondérée. Effectuer une moyenne arithmétique simple dans un tel cas conduit à de nombreuses incohérences. Par exemple, l'amélioration de la précision obtenue par une moyenne arithmétique pondérée est le gage d'une augmentation du nombre d'occurrences bien étiquetées, ce qui n'est absolument pas le cas pour la moyenne arithmétique simple. L'utilisation d'une moyenne arithmétique simple pose également des problèmes comme : faut-il calculer la performance moyenne en faisant la moyenne des performances ou en utilisant le gain moyen et le rappel moyen ?

5.3 Étude préliminaire

5.3.1 Objectifs

Comme nous l'avons dit au début de ce chapitre, pour tester et ajuster un algorithme de classification, nous avons besoin d'un critère de désambiguïsation lexicale relativement performant. D'un autre coté, pour évaluer et affiner un critère de désambiguïsation lexicale, nous avons besoin d'un ou de plusieurs algorithmes de classification efficaces.

De nombreuses études montrent que les cooccurrences constituent un bon critère pour identifier le sens d'un mot. Nous allons donc utiliser un algorithme de classification simple, de type liste de décisions, pour évaluer et affiner un critère basé sur les cooccurrences.

Les objectifs de cette étude préliminaire sont :

- de valider notre approche et nos outils par une première série de résultats positifs ;
- d'énoncer un premier critère sur lequel nous pourrons nous appuyer pour mettre au point nos différents algorithmes de classification dans le chapitre 6.

Dans le cadre de cette étude préliminaire, nous ne traitons pas l'ensemble des 60 vocables prévus pour l'étude complète, nous nous limitons simplement aux 20 noms de cette étude.

5.3.2 Critères étudiés

La série de critères que nous nous proposons d'étudier peut s'énoncer de la manière suivante :

lemme des n mots pleins qui suivent ou qui précèdent le mot cible.

```
Requête :
  cible:[lemme="@(1)" & lexie~"^[0-9]"]
  ([ems!~"(^N|^ADV|^V|^ADJ)"]* [ems~"(^N|^ADV|^V|^ADJ)"]){1,@(2)}
Apparence :
  [P:cible.fichier]-[P:cible.paragraphe]-[P:cible.position]
Référence : [P:end.lemme]
Discriminant : [P:cible.lexie]
```

Figure 5.4 – Règle correspondant à l'énoncé : « lemme des n mots pleins qui suivent le mot cible ».

```
Requête :
  ([ems~"(^N|^ADV|^V|^ADJ)"][ems!~"(^N|^ADV|^V|^ADJ)"]*){1,@(2)}
  <cible:[lemme="@(1)" & lexie~"^[0-9]"]>
Apparence :
  [P:cible.fichier]-[P:cible.paragraphe]-[P:cible.position]
Référence : [P:begin.lemme]
Discriminant : [P:cible.lexie]
```

Figure 5.5 – Règle correspondant à l'énoncé « lemme des n mots pleins qui précèdent le mot cible ».

Nous faisons varier la taille n de la demi-fenêtre afin d'observer l'impact de cette variation sur les performances du critère.

Pour simplifier et clarifier la modélisation de ce critère, nous avons choisi d'utiliser deux règles. La première permet de modéliser les demi-critères « *lemme des n mots pleins qui suivent le mot cible* » et la deuxième les demi-critères « *lemme des n mots pleins qui précèdent le mot cible.* ». C'est l'utilisation simultanée de ces deux règles qui constitue une modélisation fidèle du critère énoncé. Chaque règle aura deux paramètres. Le premier, `@(1)`, correspond au lemme du vocable étudié. Le second, `@(2)`, correspond à la taille de la demi-fenêtre. Ces paramètres permettent d'écrire une règle générique pour modéliser toute une famille de critères au lieu d'écrire une règle (deux demi-règles en l'occurrence) par vocable et par taille de demi-fenêtre.

Les figures 5.4 et 5.5 représentent les deux règles en question.

Prenons, par exemple, $n = 2$ (*i.e.* `@(2) = 2`), et intéressons-nous au occurrences du vocable ***constitution*** (*i.e.* `@(1) = constitution`). La figure 5.6 représente un extrait du fichier résultant de l'application du critère *lemme des deux mots pleins qui suivent ou qui précèdent le mot constitution.*

5.3.3 Algorithme de classification

L'algorithme de classification utilisé est du type *liste de décisions* pour sa simplicité de mise en œuvre (nous nous étendons plus amplement sur ce type d'algorithme dans la section 6.5). Comme nous ne possédons pas encore de données pour mettre au point un algorithme de classification, nous utilisons un algorithme existant présenté par Golding (1995) dans la section *Hybrid method 1 : decision lists* de son article. Il s'agit en fait du même algorithme que celui utilisé par Yarowsky (1994b) mais étendu aux classifications à plus de deux classes (nous détaillons et affinons cet algorithme dans la section 6.5.4).

La mesure utilisée par cet algorithme pour ordonner les indices dans la liste de décisions est :

$$fiabilité(indice) = \max_{lexie} p(lexie/indice)$$

A-8386-84	1	admirer	aïeul	retour	être
A-8969-36	1	débat	esprit	intéresser	ne
A-14036-230	1.1	civil	clergé	janséniste	rédiger
A-22-18	4	anné	ordinaire	taille	tempérament
A-24392-128	3	corps	lumière	nature	transparent
A-24486-2	2	esprit	profond	théorie	tout
A-24497-8	2	esprit	imaginatif	physique	théorie
A-24560-72	2	anglais	esprit	science	usage
P-997-265	3	complet	permettre	relativement	roche
A-24866-29	2	admettre	physique	théorie	être
O-5837-79	5	civil	partie	plainte	tribunal
O-5851-25	5	civil	forme	partie	respectif
⋮	⋮	⋮	⋮	⋮	⋮

Figure 5.6 – Résultat de l'application des règles des figures 5.4 et 5.5 avec `@(1)` = `constitution` et `@(2)` = `2`.

Conformément à ce que fait Golding [2], pour garantir que $p(lexie/indice) > 0$, nous lissons les données d'apprentissage en ajoutant un au nombre de fois que chaque *indice* est observé pour chaque *lexie*.

$p(lexie/indice)$ est estimé par la fraction $n_{lexie,indice}/n_{lexie}$ où $n_{lexie,indice}$ est le nombre de fois que l'indice *indice* a été rencontré pour un vocable de lexie *lexie* et n_{lexie} est le nombre d'instances de lexie *lexie* du vocable.

Pour éviter d'effectuer trop de mauvaises classifications, nous n'effectuons une classification que si $fiabilité(indice) > 0{,}5$

Les algorithmes 5.1 et 5.2 résument le fonctionnement de l'algorithme de classification pour l'étude préliminaire.

5.3.4 Résultats de l'étude préliminaire

La série des critères étudiés a été énoncée section 5.3.2 de la manière suivante :

lemme des n mots pleins qui suivent ou qui précèdent le mot cible.

La figure 5.7 représente les variations du gain et du rappel moyen pour les 20 noms en fonction de la taille n de la demi-fenêtre.

Plus la taille de la fenêtre croît, plus le rappel augmente et plus la précision, donc le gain, diminue. En effet, pour une fenêtre de deux mots pleins (donc un mot pour la taille de la demi-fenêtre) le gain est de plus de 80% et il tombe à 34% pour une taille de fenêtre de 40 mots pleins. À l'inverse, pour les mêmes tailles de fenêtre, le rappel passe respectivement de 32% à plus de 61%. Ce phénomène s'explique facilement. Plus la taille de la fenêtre croît et plus le nombre d'indices générés est grand et donc plus le nombre d'exemples sur lesquels l'algorithme peut prendre une décision est grand. Cependant, plus la taille de la fenêtre croît et plus la pertinence des indices diminue (généralement une cooccurrence éloignée du mot à désambiguïser apporte moins d'informations qu'une cooccurrence proche) et plus la précision, donc le gain, de la classification décroît.

Quelle taille de fenêtre choisir ? Tout dépend du besoin. Si l'objectif est de maximiser le gain, le plus avantageux est de ne regarder que le mot plein qui précède et celui qui suit (soit une fenêtre de plus ou moins un mot plein), si c'est le rappel qui compte, il vaut mieux augmenter la taille de la fenêtre. Pour obtenir un compromis entre gain et

2. Note de bas de page 5 page 47 de l'article de Golding (1995).

Algorithme 5.1 – Phase d'apprentissage du classifieur de l'étude préliminaire.

1: Mesurer la répartition suivant les lexies de chacun des indices des descriptions des exemples d'apprentissage. Cette étape génère un tableau à deux dimensions où une case est notée $n_{lexie,indice}$

2: Lissage : $\forall lexie, \forall indice$, $n_{lexie,indice} \leftarrow n_{lexie,indice} + 1$.

3: Ordonner les attributs en fonction de leur fiabilité pour former une liste de décisions :

$$fiabilité(indice) = \max_{lexie} p(lexie/indice)$$

4: Supprimer les indices trop peu fiables ($fiabilité(indice) \leq 0{,}5$).

Algorithme 5.2 – Phase d'exploitation du classifieur de l'étude préliminaire.

1: Parmi l'ensemble des indices de la description dont nous cherchons la classe, chercher celui qui se trouve le plus haut dans la liste de décisions.

2: **Si** aucun indice n'est trouvé **Alors**

3: Aucune classification n'est effectuée.

4: **Sinon**

5: La classification est obtenue en choisissant la classe la plus probable désignée par l'indice trouvé.

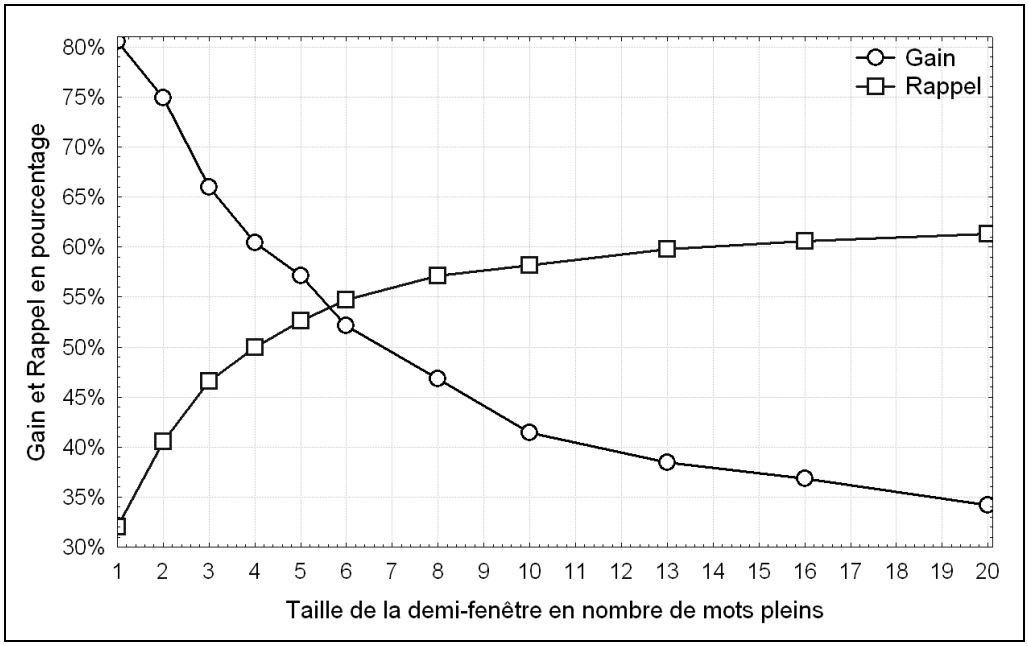


Figure 5.7 – Gain et Rappel moyen pour les 20 noms de l'étude préliminaire en fonction de la taille n de la demi-fenêtre.

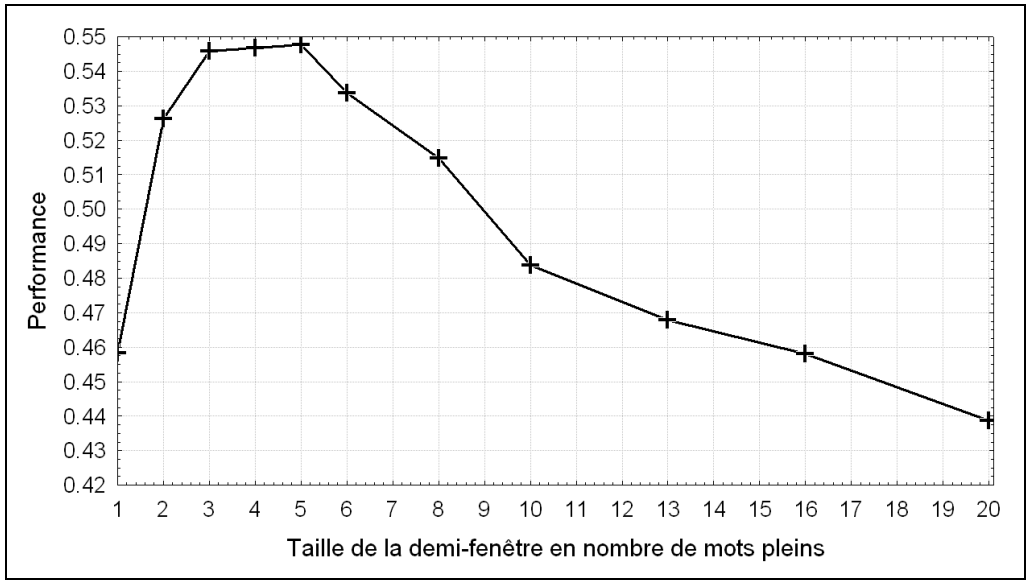


Figure 5.8 – Performance globale en fonction de la taille de la demi-fenêtre.

rappel, il faut se reporter au graphique de la mesure de performance de l'algorithme figure 5.8.

La mesure de performance combine le gain et le rappel de l'algorithme. Selon cette mesure, la taille optimale de la demi-fenêtre est de cinq mots pleins. Cependant, la performance décroît brusquement pour une taille de fenêtre plus grande. Les demi-fenêtres de trois, quatre et cinq mots pleins donnent les meilleurs résultats et constituent une sorte de palier sur le graphique. Une demi-fenêtre de quatre mots pleins semble constituer un bon compromis entre performance et robustesse.

5.3.5 Conclusion

Pour la recherche et la mise au point de nos algorithmes de classification nous choisissons donc cette taille de demi-fenêtre de quatre mots pleins. Le critère commun qui va nous permettre de comparer les algorithmes de classification dans le chapitre 6 est donc :

lemme des quatre mots pleins qui suivent ou qui précèdent le mot cible.

En plus d'énoncer un premier critère sur lequel nous pouvons nous appuyer pour mettre au point nos différents algorithmes de classification, l'objectif de cette étude préliminaire était de valider notre approche et nos outils par une première série de résultats positifs. Nous pouvons considérer que cet objectif est également rempli. Par exemple, pour une taille de demi-fenêtre de quatre mots pleins, ce premier critère et ce premier algorithme de classification permettent d'aboutir à une précision de 83% alors que la précision de l'algorithme majoritaire n'est que de 57%, ce qui correspond à un gain de 60%. Le rappel n'est cependant que de 50%. En utilisant l'algorithme majoritaire pour classer tous les exemples non classés par notre algorithme, la précision est de 69% (tout comme le rappel puisque tous les exemples sont classés) et le gain est de 27%. Ces résultats sont comparables, bien qu'en retrait, à ceux obtenus par d'autres équipes sur des corpus de langue anglaise et devraient être améliorés après la recherche d'une meilleure classification (chapitre 6) et l'étude approfondie de différents critères (chapitre 7).

Chapitre 6

Classification supervisée, théorie et pratique

6.1 Introduction

6.1.1 Généralités

Les méthodes de classification ont pour but d'identifier les classes auxquelles appartiennent des objets à partir de certains traits descriptifs. Elles s'appliquent à un grand nombre d'activités humaines et conviennent en particulier au problème de la prise de décision automatisée. Dans le cadre de la classification supervisée, la procédure de classification est produite automatiquement à partir d'un ensemble d'exemples, un exemple consistant en la description d'un cas avec la classification correspondante. Le problème est généralement un problème inductif : il s'agit d'extraire une règle générale à partir de données observées. Il existe également des méthodes d'apprentissage qui ne cherchent pas à effectuer de généralisation et donc ne constituent pas des méthodes inductives. C'est le cas des méthodes d'apprentissage basées sur les instances. Dans tous les cas, la procédure générée doit classer correctement les exemples de l'échantillon mais surtout avoir un bon pouvoir prédictif pour classer correctement de nouvelles descriptions.

Les méthodes de classification supervisée semblent tout à fait correspondre à notre problème de désambiguïsation lexicale automatique. Nous avons utilisé avec succès un algorithme issu d'une telle méthode dans notre étude préliminaire (cf. section 5.3). De nombreuses équipes de recherche utilisent des techniques de classification supervisée dans le domaine de la désambiguïsation lexicale automatique. Bien qu'il ne s'agisse pas du sujet principal de ce travail de thèse, avant de nous plonger dans l'étude de critères de désambiguïsation lexicale automatique, il nous paraît tout à fait opportun de nous intéresser aux problèmes de la classification supervisée.

6.1.2 Méthodes de classification supervisée

Il existe un grand nombre de méthodes distinctes en classification supervisée. Chaque méthode comporte des avantages, des inconvénients et des limites.

La statistique et l'analyse de données sont parmi les premières disciplines à s'être intéressées aux problèmes de classification. Ces disciplines ont étudié et proposé de multiples méthodes de classification comme, par exemple, la régression ou encore l'analyse discriminante. L'apprentissage supervisé visant à inférer, à partir d'un ensemble

limité d'exemples, des règles générales s'appliquant avec de bonnes chances de succès à une situation nouvelle, le cadre naturel de ce type de problème est celui de la théorie des probabilités. La théorie bayesienne de la décision s'inscrit dans ce cadre et a conduit, entre autres, au célèbre classifieur naïf de Bayes. Toutes ces méthodes sont basées sur des hypothèses probabilistes et sont qualifiées de méthodes paramétriques. Ces méthodes ont été développées en statistique dans les années 1920-1930, par Fischer en particulier. Lorsqu'aucune hypothèse sur la distribution des probabilités n'est faite, les méthodes sont qualifiées de non paramétriques. Les problèmes à résoudre sont alors plus complexes et les premières méthodes non paramétriques développées en statistique remontent aux années 1960. Comme exemples de méthodes non paramétriques, nous pouvons citer les méthodes des k plus proches voisins, les arbres de décision, les réseaux de neurones, les algorithmes génétiques, etc.

L'apprentissage basé sur les arbres de décision est l'une des méthodes les plus utilisées et les plus pratiques pour réaliser de l'inférence inductive. Cette méthode permet de produire des fonctions à valeur discrète (symbolique) sous forme d'arbre de décision. De manière à être plus lisibles et compréhensibles, les arbres de décisions peuvent être représentés par une liste de règles SI-ALORS. Trois systèmes ont plus particulièrement marqué les travaux sur les arbres de décision : CART (Breiman, Friedman, Olshen & Stone, 1984), ID3 (Quinlan, 1986) et C4.5 (Quinlan, 1993).

Le formalisme des listes de décision a été présenté par Rivest (1987). Des déclinaisons de ce formalisme ont ensuite largement été utilisées par Yarowsky (1993, 1994b), Golding (1995), Mooney (1996) et Wilks et Stevenson (1998), par exemple. Une liste de décisions est une liste ordonnée de formules logiques sur les attributs en forme conjonctive normale. Il s'agit d'une généralisation des arbres de décision.

L'idée du réseau de neurones formels vient de l'étude du cerveau humain. Les réseaux de neurones formels constituent une méthode robuste pour approximer des fonctions à valeur continue (réelle) ou discrète (symbolique). Pour certains types de problèmes, les réseaux de neurones font partie des techniques les plus efficaces actuellement connues.

Contrairement aux méthodes que nous venons de citer, qui construisent explicitement une procédure de classification lors de l'apprentissage, les méthodes d'apprentissage basées sur les instances (*instance-based learning*, *case-based learning* ou encore *memory-based learning* en anglais) se contentent de mémoriser chacun des exemples. Il ne s'agit donc pas de méthodes inductives. À chaque fois qu'un nouvel exemple est rencontré, sa relation avec les exemples en mémoire est examinée de manière à déterminer la classe du nouvel exemple. Parmi les méthodes d'apprentissage basées sur les instances, la plus classique est l'algorithme des k plus proches voisins.

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisements, mutations, etc. Les algorithmes génétiques ont déjà une histoire relativement ancienne puisqu'ils remontent à 1962 (Holland, 1962).

Il existe encore bien d'autres méthodes de classification supervisée. Daelemans (1999) ou encore Zavrel, Degroeve, Kool, Daelemans et Jokinen (2000) constituent de bonnes introductions aux méthodes de classification supervisée dans le domaine du traitement automatique des langues. Pour une documentation plus détaillée sur les méthodes d'apprentissage, se référer à Mitchell (1997) et à Witten et Frank (2000).

6.1.3 Présentation des sections

Dans ce chapitre, nous commençons par présenter la problématique de la classification supervisée (**section 6.2**). Nous donnons la définition formelle du problème et

discutons de l'erreur apparente et de l'estimation de l'erreur réelle.

Avant de présenter des algorithmes de classification supervisée, il reste quelques points à éclaircir comme les bornes inférieures et supérieures des performances qu'il faut prendre en considération pour des algorithmes de classification supervisée dans le cadre de la désambiguïsation lexicale. C'est, entre autres, l'objet de la **section 6.3**.

La **section 6.4** est dédiée à l'apprentissage bayesien.

La **section 6.5** est dédiée à l'apprentissage de type liste de décisions basée sur une métrique.

La **section 6.6** est consacrée à l'apprentissage basé sur les instances.

Nous synthétisons les résultats obtenus avec chacun des classifieurs évalués dans la **section 6.7**.

Ce chapitre a été écrit en s'appuyant sur le livre *machine learning* de Mitchell (1997), sur les notes du cours *apprentissage à partir d'exemples* de Denis et Gilleron (2000), sur les notes du cours *apprentissage symbolique supervisé* de Ketterlin (2001), ainsi que sur quelques articles qui sont cités au fur et à mesure.

6.2 Problématique

6.2.1 Définitions préliminaires

Le but des algorithmes de classification est, pour une instance donnée, de trouver la classe à laquelle elle appartient.

Définition 6.1 – Classe –
Les classes sont les groupes auxquels appartiennent les instances.

Dans notre cas l'ensemble des classes d'un vocable est l'ensemble des lexies de ce vocable. Les instances d'un vocable sont les apparitions de ce vocable dans un corpus.

Définition 6.2 – Attribut –
Les attributs sont les unités qui prennent une valeur donnée pour chaque instance. Ils sont fonction du domaine sur lequel opère l'algorithme de classification.

En météorologie, les attributs peuvent être la température, la pression, le pourcentage d'humidité, etc. Dans notre cas, un attribut peut, par exemple, être *le premier mot à droite de l'instance du vocable à désambiguïser*. Cependant, un problème se pose lorsque nous nous intéressons aux *trois mots qui suivent le mot à désambiguïser sans tenir compte de leur position*. En effet, le fait d'affecter chacun de ces trois mots à trois attributs distincts les différencie. Nous parlerons de ce problème de représentation des attributs dans la section 6.2.2.

Définition 6.3 – Exemple –
Un exemple consiste en la description d'un cas avec sa classification correspondante.

Dans notre cas, chaque exemple est associé à une seule classe [1]. Un exemple est une instance, dans un corpus, d'un vocable étudié accompagnée de sa description générée par un ou plusieurs critères et de sa classe d'appartenance, c'est-à-dire sa lexie.

Définition 6.4 – Description –
Une description est la description d'un cas. C'est l'affectation d'une valeur à chacun des attributs.

1. Dans le problème de désambiguïsation lexicale tel que nous le posons, chaque exemple est associé à une seule classe car chaque instance d'un vocable ne possède qu'une seule lexie. Ce n'est pas toujours le cas. Il existe des problèmes de classification dans lesquelles un exemple peut être associé, partiellement ou totalement, à plusieurs classes.

Pour nous, la description, correspondant à l'instance d'un vocable dans le corpus, est construite à partir des indices générés par l'application de un ou plusieurs critères à l'instance de ce vocable.

Définition 6.5 – Indice –
Nous appelons indices les éléments atomiques générés par les critères.

6.2.2 Comment représenter les attributs ?

L'application d'un ou plusieurs critères sur une instance d'un vocable dans le corpus génère trois types d'information :

1. la position de l'occurrence dans le corpus ;
2. la lexie de cette occurrence ;
3. l'énumération de tous les indices qui sont des instances des phénomènes linguistiques désignés par le ou les critères.

Prenons un exemple concret en considérant un extrait du tableau de la figure 5.6, lui-même extrait du résultat de l'application des règles des figures 5.4 et 5.5 :

	réference de l'exemple	*classe ou lexie*	*Énumération des indices*			
$exemple_1$	A-8386-84	1	admirer	aïeul	retour	être
$exemple_2$	A-8969-36	1	débat	esprit	intéresser	ne
$exemple_3$	A-14036-230	1.1	civil	clergé	janséniste	rédiger
$exemple_4$	A-22-18	4	année	ordinaire	taille	tempérament
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 6.1 – Extrait du tableau de la figure 5.6.

Dans le tableau de la figure ci-dessus, chaque ligne correspond à un exemple. La colonne *référence* permet d'identifier chaque exemple et de le retrouver dans le corpus. La deuxième colonne contient les classes, c'est-à-dire la lexie des exemples. Les colonnes qui suivent correspondent à l'énumération des indices générés par le ou les critères. Il est clair que c'est à partir de cette énumération que la description doit être générée. Cependant, il est difficile de statuer sur la représentation des attributs que nous devons adopter pour modéliser notre problème de désambiguïsation lexicale automatique. En effet, deux possibilités s'offrent à nous :

1. Nous pouvons considérer que cette énumération constitue la description proprement dite de l'exemple. Dans ce cas, si un critère génère n indices, le problème de classification comporte n attributs ($attribut\_1$ à $attribut\_n$). Les indices sont alors les valeurs que peuvent prendre les n attributs.
2. Nous pouvons considérer que les attributs sont les indices eux-même. Dans ce cas, le problème de classification comporte autant d'attributs que d'indices différents générés pour tous les exemples. Le nombre d'attributs n'est plus ici de l'ordre de la dizaine mais plutôt du millier, voire de la dizaine de milliers. Tous les attributs sont alors binaires : dans un exemple, l'indice a été généré ou pas.

Le critère ayant permis de générer les exemples du tableau de la figure 6.1 est (cf. section 5.3.2) :

lemme des n mots pleins qui suivent ou qui précèdent le mot cible.

Les indices de ces exemples ne sont pas ordonnés. Par exemple, dans le tableau 6.1, que l'exemple $exemple_1$ ait la description {admirer, aïeul, retour, être} ou la

description {`être`, `aïeul`, `admirer`, `retour`}, ne doit strictement rien changer à la classification de cet exemple. Si le critère tenait compte de la position des mots, nous ajouterions un préfixe au début des indices. Ainsi, pour un tel critère, si le lemme `admirer` se trouvait en première position après le mot à désambiguïser, l'indice généré serait de la forme `+1_admirer`, s'il se trouvait deux mots à gauche, l'indice généré serait de la forme `-2_admirer`.

L'inconvénient de la première représentation des attributs est que celle-ci différencie les indices (l'affectation des indices aux attributs ordonne les indices). Cette manière de procéder peut poser des problèmes avec certaines méthodes de classification. En effet, pour un algorithme de classification qui différencie les attributs, le fait que l'indice `admirer`, par exemple, soit affecté à l'attribut *attribut*_1 ou à l'attribut *attribut*_3 a de l'importance (ce comportement va à l'encontre de ce que nous désirons). Cependant, cette représentation des attributs ne pose pas de problème avec le classifieur naïf de Bayes qui ne différencie pas les attributs. C'est d'ailleurs la représentation adoptée par Yarowsky (1994a) ou encore par Ng (1997).

La seconde représentation n'est pas sujette à ce problème d'ordonnancement intrinsèque des indices. L'inconvénient de cette représentation est que les attributs sont extrêmement nombreux, mais aussi que la condition d'indépendance des attributs est violée de manière encore plus patente. Cette condition d'indépendance des attributs est nécessaire pour le classifieur naïf de Bayes (nous y reviendrons section 6.4.4). C'est cette seconde manière de procéder qui a été adoptée par Mooney (1996) qui évalue pourtant, entre autres, le classifieur naïf de Bayes.

Escudero *et al.* (2000c) ont abordé ce problème de représentation des attributs. Ils ont mené les deux alternatives de front et la première l'emporte sur la seconde dans leurs expériences. Nous choisirons l'une ou l'autre représentation en fonction de la méthode de classification étudiée.

6.2.3 Erreur de classification et classification supervisée

Pour formaliser notre propos nous utilisons les notations suivantes :

- la population, c'est-à-dire l'ensemble des occurrences à classifier, est notée Π ;
- l'ensemble des descriptions est noté D ;
- l'ensemble des classes est noté $\{1,\ldots,c\}$;
- $X : \Pi \rightarrow D$ est la fonction qui associe une description à tout élément de la population ;
- $Y : \Pi \rightarrow \{1,\ldots,c\}$ est la fonction de classement qui associe une classe à tout élément de la population ;
- $C : D \rightarrow \{1,\ldots,c\}$ est appelée fonction de classement ou procédure de classification.

Le but de l'apprentissage est de rechercher une procédure de classification C telle que $C \circ X = Y$ ou, de manière plus réaliste, telle que $C \circ X$ soit une bonne approximation de Y. Pour comparer différentes procédures de classification nous avons besoin d'une mesure appelée erreur de classification [2] :

Définition 6.6 – Erreur de classification –
Soit C une fonction de classement, l'erreur $E(d)$ pour une description d est la probabilité qu'un élément de description d de la population soit mal classé par C :

$$E(d) = P(Y(o) \neq C(d) \ / \ X(o) = d) \text{ avec } o \in \Pi.$$

2. L'*erreur de classification* est également appelée *erreur réelle* par opposition à l'*erreur apparente* (cf. définition 6.8).

L'erreur de classification $E(C)$ d'une fonction de classement est la moyenne pondérée des erreurs sur les descriptions :

$$E(C) = \sum_{d \in D} \left(E(d) \cdot P(X = d) \right).$$

Nous allons maintenant définir formellement ce qu'est la classification supervisée :

Définition 6.7 – Classification supervisée –
La classification supervisée consiste à inférer une fonction de classement dont l'erreur de classification (au sens de la définition 6.6) est minimal.

6.2.4 Leurre de l'erreur apparente

L'objectif de la classification supervisée est, à partir d'un échantillon d'exemples, d'inférer une procédure de classification dont l'erreur de classification est minimale, c'est-à-dire telle que la probabilité, que la procédure de classification classe mal un exemple tiré aléatoirement, soit minimale. Par conséquent, la procédure de classification doit avoir un bon pouvoir prédictif, c'est-à-dire qu'elle doit être capable de classer correctement de nouveaux exemples (nouveaux au sens de non présents dans l'échantillon).

Cependant, l'algorithme de classification n'a que l'échantillon pour donnée et il est tout à fait possible de générer une procédure qui classifie bien tous les exemples de l'échantillon, mais qui a un mauvais pouvoir de prédiction. Soit, par exemple, la procédure de classification suivante :

Tous les exemples de l'échantillon d'apprentissage sont mémorisés. Lorsqu'une description est présentée au système, si cette description correspond à la description d'un exemple de l'échantillon, la classe de cet exemple est retournée, sinon une classe choisie au hasard est retournée.

Figure 6.2 – Exemple de procédure minimisant l'erreur apparente mais pas l'erreur réelle.

Cette procédure fait peu d'erreurs sur les exemples de l'échantillon (aucune si tous les exemples ont des descriptions distinctes), en revanche, il est évident que son pouvoir prédictif est très mauvais.

Introduisons maintenant la définition de l'erreur apparente :

Définition 6.8 – Erreur apparente –
Soit S un échantillon et C une procédure de classification. Le taux d'erreur apparent sur S est $E_{app}(C) = \dfrac{mal\,classé}{Card(S)}$ où $mal\,classé$ est le nombre d'exemples de S qui sont mal classés par C, $Card(S)$ étant le cardinal de S.

L'objectif d'inférer une procédure de classification, dont l'erreur de classification est minimale, en se basant uniquement sur l'échantillon pose deux problèmes importants :

1. l'espace de toutes les fonctions de D dans $\{1 \dots c\}$ est généralement de taille considérable ;
2. l'erreur apparente est, en général, une version très optimiste de l'erreur réelle.

Le premier problème rend impossible d'explorer l'espace de toutes les fonctions pour des raisons de complexité en temps de calcul. Il est donc difficile de trouver une procédure de classification dont l'erreur apparente est minimale. Le second problème rend cette recherche inutile puisque minimiser l'erreur apparente n'implique pas forcément de minimiser l'erreur réelle. Ces deux difficultés nous amènent à limiter la recherche d'une fonction à un espace plus restreint. Cette restriction peut permettre d'éviter d'aboutir à des procédures trop spécialisées comme celle de la figure 6.2.

Le problème de la classification supervisée peut ainsi se réécrire : sélectionner dans un ensemble C de procédures de classification une procédure de classification C_i telle que l'erreur apparente $E_{app}(C_i)$ soit petite, tout en essayant de s'assurer que l'erreur réelle $E(C_i)$ soit également petite.

6.2.5 Estimation de l'erreur réelle

L'apprentissage à partir d'échantillons pose immédiatement la question de la pertinence de la procédure induite. Il faut donc être capable d'estimer la qualité de cette procédure induite à partir de l'échantillon. Disposer d'un ensemble permettant de tester la qualité de la procédure de classification induite résoudrait le problème.

Pour cette raison, il faut partitionner l'échantillon en un ensemble d'apprentissage S et un ensemble test T. La répartition entre les deux ensembles doit être faite aléatoirement. L'apprentissage est alors effectué sur l'ensemble S pour générer une procédure de classification C. L'estimation $\widehat{E}(C)$ de l'erreur réelle $E(C)$ est alors l'erreur apparente de C mesurée sur l'ensemble test T, soit :

$$\widehat{E}(C) = \frac{mal\,classé(T)}{card(T)}$$

où $mal\,classé(T)$ est le nombre d'éléments de T mal classés par la procédure C.

L'estimation est faite sur un ensemble indépendant de celui qui a servi à l'apprentissage, ce qui permet de supposer que l'erreur apparente sur l'ensemble test est une bonne estimation de l'erreur réelle.

La répartition de l'échantillon entre les deux ensembles se fait généralement dans des proportions de 1/2 pour chacun des deux ensembles ou bien de 2/3 pour l'ensemble d'apprentissage et de 1/3 pour l'ensemble de test.

Cependant, cette technique possède plusieurs inconvénients :

- la qualité de l'apprentissage augmente avec la taille de l'ensemble d'apprentissage, il est donc dommage de se passer de la moitié, ou du tiers, de l'échantillon pour l'apprentissage ;
- la précision de l'estimation augmente également avec la taille de l'ensemble test, il est donc dommage de se passer de la moitié, ou des deux tiers, de l'échantillon pour l'estimation de la précision ;
- parfois la taille de l'échantillon est trop petite pour pourvoir réaliser un apprentissage en se passant d'une partie de cet ensemble.

L'idéal serait de pouvoir apprendre sur tous les exemples et de pouvoir malgré tout obtenir une estimation satisfaisante du taux d'erreur. L'objectif étant toujours d'estimer l'erreur réelle $E(C)$ d'une procédure de classification C produite par un algorithme A sur un échantillon S, une solution à ce problème est apportée par la méthode de validation croisée k-fois (algorithme 6.1). Cette méthode consiste à partitionner aléatoirement l'ensemble des données en k sous-ensembles. L'apprentissage est réalisé sur $k-1$ de ces sous-ensembles et l'évaluation est réalisée sur l'ensemble retiré. Il faut réitérer cette

Algorithme 6.1 – Validation croisée k-fois.

Requiert: Un ensemble d'apprentissage S

Requiert: Un algorithme A permettant de produire une procédure de classification C

1: Partitionner S aléatoirement en k sous-ensembles $S_1, \ldots, S_k$.
2: **Pour tout** i de 1 à k **Faire**
3: appliquer A à l'échantillon $S - S_i$ et générer une procédure de classification C_i ;
4: calculer $\widehat{e}_i$ l'erreur apparente de C_i sur S_i.
5: Retourner $\widehat{E}(C) = \sum_{i=1}^{k} \frac{\widehat{e}_i}{k}$ comme estimation de $E(C)$.

opération k fois pour réaliser une évaluation sur chacun des k sous-ensembles. Cette méthode est très largement utilisée quoique ses justifications théoriques soient encore discutées en statistique. En procédant à une étude empirique, il a été établi que la meilleure stratégie est une validation croisée avec $k = 10$ (*tenfold cross validation* en anglais). Il s'agit d'un compromis entre un biais qui diminue et une variance qui s'accroît avec l'augmentation du nombre de sous-ensembles. Pour les petits ensembles il est possible de choisir $k = card(S)$. Il faut noter que la méthode est coûteuse en temps de calcul car il faut effectuer k sessions d'apprentissage, ce qui peut être rédhibitoire si le temps de calcul de l'algorithme est long.

Pour mesurer les performances de nos algorithmes de classification, nous ne travaillons pas sur une estimation de l'erreur réelle mais plutôt sur les mesures présentées section 5.2.5. L'estimation de l'erreur réelle peut être directement reliée au rappel, présenté dans la section 5.2.5, par la formule suivante :

$$Rappel(C) = 1 - \widehat{E}(C) \ .$$

Lorsque nous travaillons sur une procédure de classification qui effectue une classification de toutes les occurrences nous avons :

$$Précision(C) = Rappel(C) = Performance(C) = 1 - \widehat{E}(C) \ .$$

Bien entendu, lorsque nous parlons de la précision, du gain, du rappel ou de la performance d'un algorithme ou d'une procédure de classification, il s'agit d'un abus de langage et il faut comprendre estimation de la précision, du gain, du rappel ou de la performance.

6.3 Préalable à la mise au point de classifieurs

6.3.1 Bornes inférieures et supérieures des performances des classifieurs

Borne inférieure des performances

La borne inférieure des performances d'une méthode de désambiguïsation lexicale automatique est la précision minimum en dessous de laquelle la méthode ne doit pas descendre sous peine de ne rien apporter au niveau de la désambiguïsation. Cette borne inférieure constitue donc une référence à laquelle toute méthode de désambiguïsation

doit être comparée. En fait, il existe de nombreuses façons d'estimer cette borne inférieure.

Une manière de calculer la précision minimale d'un classifieur est de mesurer la précision obtenue en tirant la lexie au hasard pour chaque classification. Il est évident qu'un classifieur doit faire mieux qu'un simple tirage au hasard. La valeur de cette borne minimale P_{hasard}, pour un vocable donné, peut être estimée par $P_{hasard} = \frac{1}{nb\_lexies}$ où nb_lexies est le nombre de lexies du vocable. Peu d'études utilisent cette référence, et quand elles le font, c'est généralement conjointement à d'autres références.

Il est très facile de surpasser la précision P_{hasard} que nous venons d'énoncer, il suffit pour cela d'étiqueter toutes les occurrences d'un vocable avec sa lexie la plus fréquente. La précision ainsi obtenue, P_{maj} est forcément plus importante que P_{hasard} puisque nous avons toujours $P_{maj} \geq \frac{1}{nb\_lexies}$. En fait $P_{maj} = P_{hasard} = \frac{1}{nb\_lexies}$ quand toutes les lexies du vocable sont équiprobables, ce qui est extrêmement rare. Cette référence est employée dans de nombreuses études (Gale *et al.*, 1992a ; Yarowsky, 1994a ; Golding, 1995 ; Bruce *et al.*, 1996 ; Escudero *et al.*, 2000c ; etc.). C'est la référence que nous avons utilisée dans l'étude préliminaire (section 5.3) et que nous continuerons d'utiliser dans le reste de notre étude. Gale, Church et Yarowsky (1992a) pensent qu'il s'agit d'une bonne référence et qu'elle est même, dans des cas particuliers, difficile à surpasser. Prenons par exemple le vocable *chef*. Il possède 1133 occurrences dans notre corpus et 11 lexies dans notre dictionnaire. Pour chacune de ces occurrences, un classifieur devra affecter l'une de ces 11 lexies au risque de commettre une erreur. Le classifieur majoritaire, qui affecte la lexie la plus fréquente à toutes les occurrences, va réaliser 861 bonnes classifications puisque c'est la fréquence de la lexie la plus fréquente. Dans ce cas, la référence $P_{maj} = 76\%$ est bien supérieure à la référence $P_{hasard} = 9\%$ et peut s'avérer difficile à surpasser.

Dans le cadre de la campagne d'évaluation SENSEVAL (Kilgarrif & Rosenzweig, 2000), les algorithmes de référence utilisés sont plus complexes que l'algorithme majoritaire. Ils sont basés sur des variantes des algorithmes de Lesk (1986) qui sont plus performants que l'algorithme majoritaire et qui utilisent l'information de dictionnaires informatisés pour prendre leur décision.

Borne supérieure des performances

Pour mesurer la précision d'un algorithme de désambiguïsation lexicale, nous utilisons des corpus d'évaluation lexicalement étiquetés. Cet étiquetage lexical est réalisé manuellement par des linguistes ou des lexicographes. Aux erreurs de classification inhérentes à toute intervention humaine, viennent s'ajouter des fluctuations importantes dues à la subjectivité de l'annotateur. Ce désaccord entre annotateurs devient parfois aussi important que si l'affectation des lexies avait été faite au hasard (Véronis, 1998). Il est donc important d'estimer l'accord entre plusieurs annotateurs (ITA pour *InTer-annotator Agreement* en anglais) humains pour juger de la qualité d'un corpus d'évaluation. Imaginons qu'un second annotateur ne soit d'accord que sur 80% des lexies du corpus d'évaluation étiqueté par un premier annotateur. Dans ce cas, que peut signifier une précision supérieure à 80% obtenue par un classifieur ? Ainsi, l'ITA fournit une borne supérieure à la précision d'un algorithme de désambiguïsation lexicale (Kilgarriff, 1998a).

Jorgensen (1990) ainsi que Véronis (1998) montrent que l'ITA est très mauvais, et largement inférieur à 70%, en utilisant les définitions d'un dictionnaire classique. Par exemple, deux équipes indépendantes ont chacune constitué un corpus : le corpus SEMCOR (Fellbaum, 1998) et le corpus DSO (Ng & Lee, 1996). Ces deux corpus ont

été étiquetés lexicalement avec l'inventaire des sens de WordNet (Fellbaum, 1998) et comportent tous deux des extraits du corpus BROWN. Il y a donc une intersection non nulle entre les extraits de textes de ces deux corpus. Le degré d'accord entre les annotateurs de ces deux corpus sur cette intersection n'est que de 57% (Ng & Lee, 1996 ; Kilgarriff, 1998b). Véronis (1998), Gale *et al.* (1992a) et Kilgarrif et Rosenzweig (2000) montrent que, en prenant certaines mesures, il est possible d'améliorer l'ITA pour arriver jusqu'à 95% voire 96,8% d'accord. Parmi ces mesures nous trouvons l'utilisation de dictionnaires plus adaptés (comme des dictionnaires construits d'après des observations faites sur des corpus) qui simplifient la tâche de l'annotateur. Ng, Lim et Foo (1999) et Bruce et Wiebe (1998) proposent des méthodes pour augmenter l'ITA en effectuant, entre autres, des regroupements de lexies.

Dans le cadre du projet SyntSem, la qualité de l'étiquetage a été vérifiée par une méthode de sondage, dans laquelle 1% des étiquettes tirées au hasard ont été vérifiées par un deuxième annotateur, appelé « vérificateur ». Un label a été attribué à chaque étiquette parmi trois catégories :

ACCORD : le vérificateur est en accord avec le premier annotateur ;

ERREUR : le vérificateur considère que l'étiquette est purement et simplement erronée ;

INCERTITUDE : le vérificateur est incertain.

L'évaluation montre un taux d'erreur de 1,5% sur l'ensemble du corpus. Dans 2,1% des cas, il y a incertitude de la part du vérificateur. Nous pouvons donc considérer qu'il y a accord strict dans 96,4% des cas, et accord au bénéfice du doute dans 96,4% + 2,1% = 98,5% des cas. Ces résultats semblent plus que satisfaisants étant donné la difficulté de la tâche d'étiquetage des sens, déjà mise en évidence dans d'autres projets. Les résultats par corpus (tableau 6.1) montrent que l'accord est meilleur sur le corpus JOC, ce qui est compréhensible vu la nature administrative des textes et leur thématique plus restreinte.

	ABU	JOC	MON	OUV	PER	**Total**
ACCORD	95,3%	99,2%	95,3%	96,0%	96,0%	96,4%
INCERTITUDE	2,4%	0,0%	2,4%	3,0%	3,0%	2,1%
ERREUR	2,4%	0,8%	2,4%	1,0%	1,0%	1,5%

Tableau 6.1 – ITA sur les cinq sous-corpus du projet SyntSem.

6.3.2 Critère de base et sous-corpus pour l'affinage et l'évaluation des classifieurs

Dans les sections qui suivent, nous allons présenter, implémenter, tester, et affiner quelques algorithmes de classification. Ces algorithmes nous permettront ensuite de mesurer les performances des critères que nous étudierons. Cependant, pour tester et affiner un algorithme de classification, nous avons besoin d'un critère sur lequel il puisse opérer. Pour cela, nous utilisons le critère retenu dans l'étude préliminaire (section 5.3) :

lemme des quatre mots pleins qui suivent ou qui précèdent le mot cible.

Les algorithmes de classification comportent généralement des paramètres. Nous appelons *affinage d'un algorithme* la recherche d'une valeur de paramètres permettant d'obtenir de bons résultats. Nous ne cherchons pas les paramètres optimaux qui le seraient pour notre corpus, mais pas nécessairement pour un autre. Toutefois, entre la valeur optimale des paramètres et des valeurs prises au hasard, les performances peuvent

être très différentes. Nous cherchons donc simplement des paramètres qui permettent de bien tirer parti des algorithmes étudiés.

Ajuster les paramètres de nos algorithmes de classification en utilisant une méthode de validation croisée k-fois sur l'ensemble de notre corpus pose un problème. En effet, nous allons ensuite utiliser ces algorithmes sur ce même corpus, pour notre étude des critères de désambiguïsation lexicale (chapitre 7), ce qui entachera nos résultats d'une incertitude difficilement acceptable : la qualité de la désambiguïsation n'est-elle pas biaisée par le fait que les algorithmes ont déjà rencontré les mêmes données durant la phase d'affinage des paramètres ? En toute rigueur, nous devrions ajuster les paramètres des algorithmes de classification et effectuer notre étude des critères sur deux corpus distincts. Cependant, nous priver d'une partie du corpus dans le cadre de notre étude des critères serait dommage. En effet, 53% des lexies de notre étude comportent moins de dix occurrences dans le corpus et 37% en comportent moins de quatre ! C'est pourquoi nous préférons :

1. utiliser l'intégralité de notre corpus pour mener notre étude des critères de désambiguïsation lexicale dans le chapitre 7 ;
2. utiliser seulement un dixième du corpus complet pour ajuster les paramètres de nos algorithmes dans ce chapitre.

Parmi les algorithmes de classification que nous projetons d'évaluer, deux sont du type k plus proches voisins (cf. section 6.6). Ces algorithmes nécessitent de déterminer une constante k correspondant au nombre de voisins à prendre en compte. La valeur optimale de ce paramètre k peut varier fortement en fonction des corpus, mais aussi en fonction du nombre d'occurrences de chaque classe (*i.e.* lexie) dans le corpus d'apprentissage. Ainsi, si nous prenons un dixième du corpus au hasard, nous divisons, en moyenne, par dix le nombre d'occurrences de chaque classe. La constante k ainsi déterminée risque de ne plus être pertinente sur l'ensemble du corpus dans lequel les effectifs de chaque classe seront dix fois plus importants. Aussi choisissons-nous de composer notre sous-corpus non pas en sélectionnant au hasard un dixième du corpus complet, mais plutôt en sélectionnant un dixième de nos vocables, c'est-à-dire deux noms, deux adjectifs et deux verbes. Nous avons choisi, comme représentants, les vocables *compagnie*, *degré*, *populaire*, *utile*, *connaître* et *entrer* en raison de leur mesure de fréquence d'entropie et de perplexité proches des mesures moyennes de leur catégorie grammaticale respective (cf. tableaux 4.4, 4.5 et 4.6 pages 96, 97 et 98). Le sous-corpus ainsi constitué représente 8,6% du corpus complet. L'utilisation de ce sous-corpus devrait nous permettre de limiter fortement le biais qui résulterait de l'utilisation de l'intégralité du corpus pour ajuster les paramètres de nos algorithmes.

6.3.3 Choix des algorithmes de classification

Nous ne cherchons pas à effectuer une étude comparative des différentes techniques de classification supervisée. Notre étude porte sur les critères de désambiguïsation lexicale automatique. Pour mener à bien cette étude, il nous faut cependant ajuster à notre tâche au moins un algorithme de classification. Pour que le choix du classifieur ne biaise pas trop nos résultats, il est préférable de réaliser notre étude en nous basant sur deux ou trois classifieurs différents.

Afin de mieux maîtriser et appréhender le fonctionnement des classifieurs que nous utilisons, nous avons choisi d'implémenter nous-même chacun des classifieurs utilisés. Par exemple, pour le classifieur naïf de Bayes, ce choix nous permet de définir la meilleure façon d'estimer les probabilités (cf. section 6.4.4), ce qui ne serait pas possible si nous avions choisi d'utiliser l'implémentation du projet WEKA (Witten & Frank, 2000)

de ce classifieur. Ce choix nous permet également d'adapter le code à notre convenance, de manière à observer, par exemple, pourquoi telle instance a été classée de telle façon. Pour cette raison, nous n'avons testé que des classifieurs pour lesquels nous avons pu nous procurer une documentation suffisante en temps utile (ce qui n'a pas été le cas pour le classifieur C4.5 de Quinlan par exemple).

Notre choix s'est porté sur le classifieur naïf de Bayes car il s'agit d'un classifieur simple, très largement utilisé et obtenant des résultats souvent proches des meilleurs algorithmes de classification.

Le classifieur du type liste de décisions basée sur une métrique (proche de celui utilisé par Yarowsky) s'est imposé à nous pour plusieurs raisons. La première est sa grande simplicité de mise en œuvre et son efficacité reconnue. La seconde tient au fait que ce classifieur se démarque nettement du classifieur naïf de Bayes car il ne combine pas l'information des attributs et qu'il peut traiter des attributs fortement dépendants. La dernière, et peut-être la plus importante pour nous, est la transparence de la prise de décision à travers la liste de décisions. Nous exploiterons cette transparence à plusieurs reprises au cours de notre étude des critères de désambiguïsation lexicale automatique.

Nous nous sommes enfin intéressé à des classifieurs basés sur les instances qui se démarquent nettement des deux classifieurs précédents et qui obtiennent généralement de bons résultats.

6.3.4 Application AlgoWSD pour l'implémentation des classifieurs

Nous ne nous étendrons pas sur cette application qui n'est pas destinée à être diffusée et qui ne sert que ce travail de thèse. AlgoWSD est une application orientée *ligne de commande* qui permet de réaliser des apprentissages et des évaluations d'algorithmes de classification. Nous avons mis en œuvre un mécanisme d'abstraction de l'objet classifieur de manière à cloisonner l'implémentation de chacun des classifieurs et la procédure de lecture des données d'apprentissage et d'évaluation des classifieurs.

6.3.5 Classifieur majoritaire (référence)

Présentation

Comme nous l'avons dit dans la section précédente (6.3.1), le classifieur majoritaire est employé dans de nombreuses études pour estimer la limite inférieure des performances (Gale *et al.*, 1992a ; Yarowsky, 1994a ; Golding, 1995 ; Bruce *et al.*, 1996 ; Escudero *et al.*, 2000c ; etc.). Ainsi, tout algorithme doit au moins dépasser le pouvoir prédictif de l'algorithme majoritaire qui associe à toute description la classe la plus fréquente (définition 5.3). Cet algorithme sert donc de standard de référence pour les autres algorithmes, et plus précisément de limite inférieure à leur performance. Il n'y a rien à ajuster sur cet algorithme qui ne tient compte ni des descriptions des exemples d'apprentissage ni des descriptions des exemples d'évaluation.

La phase d'apprentissage du classifieur majoritaire (algorithme 6.2) est extrêmement simple et se résume à trouver et mémoriser la classe la plus fréquente dans le corpus d'apprentissage.

Quand une nouvelle description est présentée à cet algorithme, il retourne, quelle que soit la description, la classe la plus fréquente (algorithme 6.3).

Algorithme 6.2 – Phase d'apprentissage du classifieur majoritaire.

Requiert: Un ensemble d'apprentissage S

1: $\forall$ *classe* $f_{classe} \leftarrow 0$

2: **Pour tout** $exemple \in S$ **Faire** /* Génération du vecteur f de fréquence des classes ; un *exemple* est une instance dans le corpus d'un vocable étudié accompagnée de sa description générée par un ou plusieurs critères et de sa *classe* d'appartenance. */

3: $f_{classe} \leftarrow f_{classe} + 1$ où *classe* est la classe de *exemple*.

4: $majoritaire \leftarrow \underset{classe}{\text{argmax}}\ f_{classe}$ /* La fonction *argmax* prend une variable et une expression arithmétique comme argument et retourne la valeur de la variable pour laquelle l'expression est maximale. */

Algorithme 6.3 – Phase d'exploitation du classifieur majoritaire.

Requiert: la classe majoritaire *majoritaire* déterminée pendant la phase d'apprentissage

1: **Retourner** *majoritaire*.

Noms	**Précision**	**Adjectifs**	**Précision**	**Verbes**	**Précision**
barrage	76,1%	biologique	89,9%	arrêter	23,9%
chef	76,0%	clair	29,3%	comprendre	32,6%
communication	40,6%	correct	53,4%	conclure	45,5%
compagnie	71,4%	courant	90,0%	conduire	38,2%
concentration	45,1%	exceptionnel	53,1%	**connaître**	40,1%
constitution	50,0%	frais	36,4%	couvrir	33,3%
degré	58,6%	haut	25,0%	**entrer**	26,6%
détention	72,3%	historique	87,4%	exercer	59,5%
formation	63,8%	plein	17,1%	importer	27,6%
lancement	79,7%	**populaire**	47,9%	mettre	42,2%
observation	86,0%	régulier	32,6%	ouvrir	26,0%
organe	38,3%	sain	40,3%	parvenir	36,7%
passage	37,0%	secondaire	53,8%	passer	15,8%
pied	37,6%	sensible	29,9%	porter	29,4%
restauration	43,3%	simple	41,3%	poursuivre	36,2%
solution	93,3%	strict	45,5%	présenter	40,1%
station	32,0%	sûr	45,9%	rendre	46,4%
suspension	61,8%	traditionnel	89,5%	répondre	78,3%
vol	40,3%	**utile**	42,9%	tirer	28,9%
économie	49,1%	vaste	42,4%	venir	24,9%
Moyenne	57,3%	**Moyenne**	46,4%	**Moyenne**	37,2%
Moyenne sur les 60 **vocables**			42,9%		

Tableau 6.2 – Précision de l'algorithme MAJ sur les 60 vocables. Les six vocables, qui constituent le sous-corpus utilisé pour ajuster et évaluer les classifieurs, figurent en gras.

Performances

Nous nommerons par la suite MAJ l'algorithme 6.3. Les performances de l'algorithme majoritaire sont présentées dans le tableau 6.2. La précision indiquée en face de chacune des entrées est donc la précision de l'algorithme majoritaire (définition 5.4) qui sert au calcul du gain (définition 5.5) pour les autres algorithmes.

6.4 Apprentissage bayesien

6.4.1 Généralités

Les algorithmes d'apprentissage bayesien, comme le classifieur naïf de Bayes, constituent parfois l'approche la plus pratique et la plus performante. Mooney (1996) a réalisé une étude comparative de sept algorithmes de classification pour une tâche de désambiguïsation lexicale. Il compare dans son étude les algorithmes suivants :

- le classifieur naïf de Bayes ;
- la méthode des k-plus proches voisins ;
- un perceptron à simple couche pour représenter les réseaux de neurones ;
- l'algorithme C4.5 de Quinlan (1993) pour représenter les arbres de décision ;
- trois algorithmes inductifs basés sur des règles logiques dont un qui correspond à un algorithme de type liste de décisions.

L'algorithme jugé le plus performant lors de cette étude est le classifieur naïf de Bayes. D'un autre côté, Ng (1997) compare un algorithme d'apprentissage basé sur les instances (*i.e.* basé sur des notions de proximité) sophistiqué et performant (PEBLS [3]) et constate que cet algorithme dépasse difficilement les performances du classifieur naïf de Bayes. Ce classifieur est également utilisé avec succès dans d'autres travaux, comme ceux de Leacock, Chodorow et Miller (1998).

Le classifieur naïf de Bayes constitue un algorithme incontournable dans le cadre de notre étude. Il est conditionné par une supposition forte sur l'indépendance des attributs. Cette hypothèse permet à cet algorithme de combiner simplement l'information apportée par chacun des attributs d'un exemple. Combiner l'information apportée par tous les attributs, plutôt que de se focaliser sur un seul comme le ferait un algorithme basé sur une liste de décisions, est le point fort de cet algorithme. C'est également son point faible car cette condition d'indépendance est rarement respectée (Bruce & Wiebe, 1994b). En outre, plus nous combinons des critères, plus cette condition d'indépendance est violée. Toutefois, même quand la condition d'indépendance n'est pas remplie, ce classifieur donne généralement de bons résultats (Domingos & Pazzani, 1997).

6.4.2 Rappels de probabilité

Soient A et B deux événements quelconques d'un ensemble fondamental Ω muni d'une loi de probabilité P. Nous nous intéressons à ce que devient la probabilité de A lorsque nous apprenons que B est déjà réalisé, c'est-à-dire lorsque nous restreignons l'ensemble des résultats possibles Ω à B.

3. Cet algorithme est présenté section 6.6.3.

Définition 6.9 – Probabilité conditionnelle –
La probabilité conditionnelle de A, sachant que l'événement B, de probabilité non nulle, est réalisé, est notée $P(A/B)$ et est définie par la relation suivante :

$$P(A/B) = \frac{P(A \cap B)}{P(B)}$$

Cette relation générale pour tout espace probabilisé s'interprète facilement dans le cas où Ω est un espace équiprobable, mais reste valable pour un espace non équiprobable.

En reprenant l'équation de la définition 6.9 nous déduisons immédiatement le théorème de la multiplication.

Théorème 6.1 – Multiplication –
Soient A et B deux événements quelconques, mais de probabilité non nulle, d'un ensemble fondamental Ω muni d'une loi de probabilité P. La probabilité que les événements A et B surviennent simultanément est :

$$P(A \cap B) = P(A/B)P(B) = P(B/A)P(A)$$

L'équation de la définition 6.1 peut se généraliser facilement. Soient $A_1, \ldots, A_n$ des événements quelconques d'un espace probabilisé. À partir de l'équation de la définition 6.1 nous pouvons déduire que :

$$P(A_1 \cap A_2 \ldots \cap A_n) = P(A_1)P(A_2/A_1)P(A_3/(A_1 \cap A_2))P(A_n/(A_1 \cap A_2 \ldots \cap A_{n-1}))$$

Deux événements sont indépendants si la probabilité pour que A soit réalisé n'est pas modifiée par le fait que B se soit produit, c'est-à-dire : $P(A/B) = P(A)$.

Définition 6.10 – Événements indépendants –
A et B sont deux événements indépendants si et seulement si :

$$P(A \cap B) = P(A) \cdot P(B)$$

ou encore si et seulement si $P(B/A) = P(B)$ (l'information sur la réalisation de A n'apporte rien à l'événement B) ou $P(A/B) = P(A)$.

Ce qui est défini précédemment est l'indépendance de deux événements. En considérant n événements $A_1, \ldots, A_n$, nous pouvons dire que ces n événements sont indépendants si et seulement si les deux conditions suivantes sont vérifiées :

1. ils sont tous indépendants deux à deux ;
2. $P\left(\bigcap_{i=1}^{n} A_i\right) = \prod_{i=1}^{n} P(A_i)$

A et B sont indépendants uniquement si nous pouvons prouver formellement que $P(A \cap B) = P(A) \cdot P(B)$. En pratique (*i.e* sur des valeurs numériques), nous ne pouvons pas induire l'indépendance à partir de cette égalité constatée numériquement. Nous ne pouvons que supposer très probable cette indépendance.

Remarque : attention à ne pas confondre événements indépendants et événements incompatibles !

6.4.3 Aspect théorique

Théorème de Bayes

Ce théorème se déduit de l'équation de la définition 6.1.

Théorème 6.2 – Théorème de Bayes –
Soient A et B deux événements quelconques, mais de probabilité non nulle, d'un ensemble fondamental Ω muni d'une loi de probabilité P. Le théorème de Bayes est :

$$P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$$

Classifieur de Bayes

Nous notons $P(c)$ la probabilité *a priori* qu'un exemple soit de classe c avant d'observer les données d'apprentissage. Si nous n'avons pas une telle probabilité, nous pouvons toujours assigner une probabilité *a priori* égale pour toutes les hypothèses.

De la même manière, nous notons $P(D)$ la probabilité *a priori* qu'un exemple de description D survienne. $P(D/c)$ représente la probabilité d'observer un exemple de description D quand la classe est c ou, autrement dit, $P(D/c)$ représente la vraisemblance de la description D sachant que la classe est c.

Ce que nous recherchons est la probabilité $P(c/D)$ que la classe c soit la classe d'un exemple de description D. $P(c/D)$ est appelée probabilité de c *a posteriori* car elle reflète notre confiance de voir un exemple de classe c sachant que sa description est D.

Le théorème de Bayes est la pierre angulaire des méthodes d'apprentissage bayesien car il donne le moyen de calculer la probabilité *a posteriori* $P(c/D)$ en fonction de la probabilité *a priori* $P(c)$ et des probabilités $P(D)$ et $P(D/c)$.

D'après l'équation de la définition 6.2 nous pouvons écrire :

$$\underbrace{P(c/D)}_{\substack{\text{probabilité} \\ \textit{a posteriori}}} = \underbrace{P(c)}_{\substack{\text{probabilité} \\ \textit{a priori}}} \times \underbrace{\frac{P(D/c)}{P(D)}}_{\substack{\text{rapport de} \\ \text{vraisemblance}}}$$

Dans la plupart des scénarios de classification, l'objectif est de trouver, étant donnée une description D, la classe $c \in C$ la plus probable parmi un ensemble de classes C. Une telle classe de probabilité maximale *a posteriori* est notée c_{MAP}. Nous pouvons la déterminer en utilisant le théorème de Bayes pour calculer les probabilités *a posteriori* de chacune des classes :

$$\begin{aligned} c_{MAP} &\equiv \underset{c \in C}{\operatorname{argmax}}\ P(c/D) \\ &= \underset{c \in C}{\operatorname{argmax}}\ \frac{P(D/c) \cdot P(c)}{P(D)} \end{aligned} \tag{6.1}$$

Le terme $P(D)$ dans la dernière ligne n'est pas indispensable car $P(D)$ est une constante indépendante de c. La fonction *argmax* prend une variable et une expression arithmétique comme argument et retourne la valeur de la variable pour laquelle l'expression est maximale.

Définition 6.11 – Classe de probabilité maximale *a posteriori* –
Étant donnée une description D, la classe $c_{MAP} \in C$ de probabilité maximale a posteriori*, parmi un ensemble de classes C, se calcule de la manière suivante :*

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\ P(D/c) \cdot P(c)$$

Classifieur du maximum de vraisemblance

Parfois nous supposons que toutes les classes c sont équiprobables *a priori* :

$$\forall c_i \in C \ , \forall c_j \in C \ , P(c_i) = P(c_j).$$

Dans ce cas, nous pouvons encore simplifier l'équation 6.1 en ne considérant que le terme $P(D/c)$ pour trouver la classe la plus probable. Cette classe qui maximise $P(D/c)$ est appelée classe de maximum de vraisemblance et notée c_{MV} :

Définition 6.12 – Classe de maximum de vraisemblance –
Étant donnée une description D, la classe $c_{MV} \in C$ de maximum de vraisemblance, parmi un ensemble de classes C, se calcule de la manière suivante :

$$c_{MV} \equiv \underset{c \in C}{\operatorname{argmax}} P(D/c)$$

Classifieur naïf de Bayes

Une méthode très pratique et très utilisée de l'apprentissage bayesien est le classifieur naïf de Bayes. Dans certains domaines, ses performances rivalisent avec celles de méthodes de classifications basées sur des réseaux de neurones ou des arbres de décision.

Nous notons $a_1,a_2,\ldots,a_n$ l'ensemble des attributs qui composent une description D. L'approche bayesienne pour classer un nouvel exemple est de chercher la classe la plus probable, c_{MAP}, étant donnée la description $D = \{a_1,a_2,\ldots,a_n\}$ qui décrit cet exemple.

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} \ P(c_j/a_1,a_2,\ldots,a_n)$$

Nous pouvons utiliser le théorème de Bayes pour réécrire cette expression qui devient :

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} \ P(a_1,a_2,\ldots,a_n/c_j)P(c_j) \tag{6.2}$$

Il faut maintenant estimer les deux termes de l'équation 6.2 en se basant sur l'ensemble d'apprentissage. Il est facile d'estimer le terme $P(c_j)$ en dénombrant simplement les fréquences de chacune des classes c_j dans l'ensemble d'apprentissage. Cependant, estimer les différents termes $P(a_1,a_2,\ldots,a_n/c_j)$ n'est pas faisable à moins de posséder un ensemble d'apprentissage extraordinairement grand. En effet, le nombre de ces termes est égal au nombre de descriptions possibles $a_1,a_2,\ldots,a_n$ multiplié par le nombre de classes (card(C)). De plus il faut observer chacune de ces descriptions un nombre suffisant de fois pour obtenir des probabilités fiables.

Le classifieur naïf de Bayes est basé sur la supposition d'indépendance conditionnelle des attributs pour une classe donnée. Dans ce cas, d'après l'équation de la définition 6.10, la probabilité d'observer la conjonction $a_1,a_2,\ldots,a_n$ sachant la classe c_j se réduit au produit des probabilités des attributs pris individuellement :

$$P(a_1,a_2,\ldots,a_n/c_j) = \prod_i P(a_i/c_j)$$

En substituant ce résultat dans l'équation 6.2, nous obtenons la méthode de classification naïve de Bayes. Nous notons c_{NB} la classe retournée par cette classification.

Définition 6.13 – Classe de Bayes naïf –
Étant donnée une description D composée des attributs $a_1,a_2,\ldots,a_n$, la classe $c_{NB} \in C$, parmi un ensemble de classes C, retournée par le classifieur naïf de Bayes se calcule de la manière suivante :

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \ P(c_j) \prod_i P(a_i/c_j)$$

Nous supposons ici que les attributs a_1,a_2,...,a_n sont conditionnellement indépendants.

Le nombre de termes $P(a_i/c_j)$ à estimer est égal au produit du nombre d'attributs par le nombre de classes. Ce nombre est très largement inférieur au nombre de termes $P(a_1,a_2,\ldots,a_n/c_j)$ que nous devions calculer précédemment.

En résumé, la méthode d'apprentissage naïve de Bayes implique une phase d'apprentissage pendant laquelle les différentes probabilités $P(c_j)$ et $P(a_i/c_j)$ sont estimées en se basant sur leur fréquence dans l'ensemble d'apprentissage. Ces estimations correspondent aux hypothèses d'apprentissage. Ces hypothèses sont ensuite exploitées pour classer chaque nouvel exemple en appliquant la formule de la définition 6.13. Dans le cas où la supposition sur l'indépendance conditionnelle est exacte, le classifieur naïf de Bayes c_{NB} est identique au classifieur de probabilité *a posteriori* maximale c_{MAP}.

Estimation des probabilités

Généralement, la probabilité d'un événement e est estimée par la fraction du nombre n_e de fois que l'événement est observé sur le nombre n de fois que l'événement pourrait être observé :

$$p(e) = \frac{n_e}{n} \tag{6.3}$$

Cette estimation est satisfaisante dans bien des cas. Cependant, quand le nombre n_e devient petit et encore plus quand il devient nul, cette estimation n'est plus acceptable. D'autant plus que la méthode naïve de Bayes, formule de la définition 6.13, fait intervenir le produit $\prod_i P(a_i/c_j)$ qui devient nul si l'estimation de l'une des probabilités $P(a_i/c_j)$ est nulle. Il faut donc avoir recours à une autre méthode d'estimation des probabilités. L'une de ces méthodes est d'utiliser la m-estimation qui est une approche bayesienne de l'estimation de probabilité. Pour plus d'informations sur la m-estimation et sur les valeurs possibles de m, se référer à l'article de Cussens (1993), au livre de Mitchell (1997) page 179, ou encore au livre de Bishop, Fienberg et Holland (1975) page 407.

Définition 6.14 – m-estimation –
La m-estimation de la probabilité $p(e)$ d'un événement e se fait de la manière suivante

$$p(e) = \frac{n_e + m \cdot p}{n + m}$$

où n_e est le nombre de fois que l'événement e a été observé, n est le nombre de possibilités qu'il soit observé, p une estimation de la probabilité a priori *qu'il soit observé et m une constante appelée* équivalent de la taille de l'échantillon. *La constante m détermine l'importance accordée à p.*

En l'absence de toute information relative à p, nous supposons une répartition uniforme des probabilités *a priori*. Ainsi, si un attribut a k valeurs possibles, nous supposons $p = \frac{1}{k}$. En choisissant $m = 0$ nous retrouvons l'équation 6.3. Si $m \neq 0$ l'équation de la définition 6.14 peut être vue comme l'interprétation de la probabilité des n exemples observés augmentée de m exemples virtuels dont la distribution est donnée par p. La valeur de m donnant les meilleurs résultats est dépendante du domaine et doit être calculée expérimentalement. Nous pouvons essayer, entre autres, les valeurs $m = k$ (dans ce cas $m \cdot p = k \cdot \frac{1}{k} = 1$, ce lissage est connu sous le nom de lissage de Laplace), $m = 1$, ou encore une valeur de m dépendante des données $m = \sqrt{n}$.

6.4.4 Classifieur naïf de Bayes

Choix des attributs

Dans la section 6.2.2, nous avons posé le problème de représentation des attributs. Le classifieur naïf de Bayes ne différencie pas les attributs. Nous pouvons donc l'utiliser pour notre problème de classification avec les deux représentations des attributs.

Ce classifieur détermine la classe d'une nouvelle instance de la manière suivante (cf. équation de la définition 6.13) :

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}}\ P(c_j) \prod_i P(a_i/c_j)$$

Cette formule suppose que les attributs a_1,a_2,...,a_n soient conditionnellement indépendants. Quel que soit notre choix de représentation des attributs, cette condition est violée car les événements sont forcément plus ou moins dépendants (en fonction du choix du ou des critères). En effet, nous ne pouvons ni prouver que les attributs a_1,a_2,...,a_n sont tous indépendants deux à deux, ni prouver que $P\left(\bigcap_{i=1}^{n} a_i\right) = \prod_{i=1}^{n} P(a_i)$. Cependant, si nous choisissons la deuxième représentation des attributs, nous obtenons des milliers d'attributs binaires dont seuls quelques-uns peuvent être vrais en même temps. La probabilité de rencontrer tous les attributs dans un exemple est donc forcément nulle, ce qui implique $P\left(\bigcap_{i=1}^{n} a_i\right) = 0$ alors que $\prod_{i=1}^{n} P(a_i) \neq 0$. La condition d'indépendance des attributs est ainsi violée de manière flagrante en utilisant la deuxième représentation des attributs. Contrairement à Mooney (1996), et conformément à Yarowsky (1994a) ou Ng (1997), nous choisissons la première représentation des attributs.

Algorithmes

Soit un exemple E dont nous désirons déterminer la lexie la plus probable l_E, parmi un ensemble de lexies L, en nous basant sur la description D de E composée d'un certain nombre d'indices $D = \{i_1 \ldots i_n\}$. Soit A l'ensemble des indices des exemples d'apprentissage. En appliquant la formule de la définition 6.13 nous obtenons :

$$l_E = \underset{l_j \in L}{\operatorname{argmax}}\ P(l_j) \prod_{i_k \in D \cap A} P(i_k/l_j)$$

L'estimation des probabilités $P(l_j)$ et $P(i_k/l_j)$ se fait sur les exemples d'apprentissage.

L'estimation de $P(l_j)$ se fait tout simplement en utilisant la formule 6.3 :

$$P(l_j) = \frac{n_j}{n}$$

où n_j est le nombre d'exemples d'apprentissage dont la lexie du vocable étudié est l_j et où n est le nombre d'exemples d'apprentissage du vocable étudié.

En revanche, pour estimer les probabilités $P(i_k/l_j)$, nous utilisons une m-estimation en raison des valeurs de i_k faibles, et parfois nulles. En utilisant l'équation de la définition 6.14, nous obtenons :

$$P(i_k/l_j) = \frac{n_{kj} + m \cdot p_{kj}}{n_j + m}$$

où

- n_{kj} est le nombre d'exemples d'apprentissage dont la description contient l'indice i_k et dont la lexie du vocable étudié est l_j ;

- n_j est le nombre d'exemples d'apprentissage dont la lexie du vocable étudié est l_j ;
- p_{kj} est une estimation *a priori* de la probabilité recherchée ; comme nous ne connaissons pas cette probabilité, nous supposons une répartition uniforme des probabilités et nous posons $p_{kj} = \frac{1}{\text{card}(A)}$ où A est l'ensemble des indices des exemples d'apprentissage ;
- m est une constante, appelée « équivalent de la taille de l'échantillon », à déterminer.

La phase d'apprentissage du classifieur « naïf de Bayes » est décrite par l'algorithme 6.4.

La phase d'exploitation du classifieur « naïf de Bayes » est décrite par l'algorithme 6.5. Dans cet algorithme, nous avons ajouté un paramètre qui permet de définir un seuil de confiance en dessous duquel l'algorithme ne prend pas de décision.

Affinage et performances

Affiner cet algorithme consiste à déterminer la constante m. Nous avons choisi d'essayer plusieurs valeurs fixes de m et deux valeurs dépendantes des données $m = \sqrt{f_{lexie}}$ et $m = \text{card}(exemple)$.

La figure 6.3 montre la performance moyenne pour les 6 vocables obtenue par l'algorithme en fonction de différentes valeurs de m. Dans ces expériences $p_{min} = 0$, l'algorithme classifie donc le maximum d'exemples possible. La valeur de m fixe permettant d'obtenir les meilleures performances se situe autour de 80, ce qui donne une performance de 0,396. Les performances obtenues pour la valeur $m = \text{card}(exemple)$ dépendante des données sont mauvaises : nous écartons donc cette valeur de m. Les performances obtenues pour la valeur $m = \sqrt{f_{lexie}}$ dépendante des données sont moins bonnes que pour $m = 80$, mais sont tout de même assez proches. En effet, la performance est moins bonne de seulement 0,010, la chute de la précision est de 0,5% et celle du rappel de 0,5% également (pratiquement toutes les occurrences sont classées). Avec d'autres critères ou d'autres vocables, il est probable que la valeur de m fixe permettant d'obtenir les meilleures performances ne soit plus la même. Choisir une valeur de m dépendante des données, plutôt qu'une constante déterminée expérimentalement, nous paraît donc une meilleure solution.

Nous nommons NB(0,00) l'algorithme 6.5 avec $m = \sqrt{f_{lexie}}$ et $p_{min} = 0$. Le tableau 6.3 résume les performances moyennes de l'algorithme NB(0,00) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

Si nous ne cherchons pas à maximiser le rappel, il est possible d'améliorer la performance, c'est-à-dire la moyenne harmonique du rappel et du gain, en augmentant le seuil de confiance p_{min}. La figure 6.4 montre l'influence de la valeur du seuil p_{min}, sur la performance moyenne pour les 6 vocables. En fait, plus le seuil p_{min} est haut, jusqu'à une limite proche de 1, meilleure est la performance. Cependant, être trop proche de 1 se traduit évidemment par une chute brutale du rappel (le rappel devient même nul pour une valeur de $p_{min} = 1$) et donc des performances.

Une valeur de $p_{min} = 0{,}98$ nous semble être un bon compromis. Nous nommons NB(0,98) l'algorithme 6.5 avec $m = \sqrt{f_{lexie}}$ et $p_{min} = 0{,}98$. Le tableau 6.4 résume les performances moyennes de l'algorithme NB(0,98) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

Comme nous l'avons dit dans la section 5.2.5, page 107, il est facile de se ramener à un algorithme qui effectue une classification sur toutes les instances à classer en affectant à toutes les instances non classées la classe majoritaire. Dans ce cas de

Algorithme 6.4 – Phase d'apprentissage du classifieur naïf de Bayes.

Requiert: Un ensemble d'apprentissage S

1: $\forall\ indice\ \forall\ classe\ tab_{indice,classe} \leftarrow 0$
2: $\forall\ classe\ f_{classe} \leftarrow 0$
3: **Pour tout** $exemple \in S$ **Faire** /* Génération du tableau *tab* d'apprentissage et du vecteur f des fréquences des classes ; un *exemple* est une instance dans le corpus d'un vocable étudié accompagnée de sa description générée par un ou plusieurs critères et de sa *classe* d'appartenance. m est une constante à fournir à l'algorithme. */
4: $f_{classe} \leftarrow f_{classe} + 1$ où $classe$ est la classe de $exemple$.
5: **Pour tout** $indice \in exemple$ **Faire**
6: $tab_{indice,classe} \leftarrow tab_{indice,classe} + 1$ où $classe$ est la classe de $exemple$
7: $p \leftarrow \frac{1}{\text{nombre d'indices}}$ /* Estimation de la probabilité *a priori* */
8: **Pour tout** $classe$ **Faire** /* Estimation des probabilités */
9: $P(classe) \leftarrow \frac{f_{classe}}{\text{card}(exemple)}$
10: **Pour tout** $indice$ **Faire**
11: $P(indice/classe) \leftarrow \frac{tab_{indice,classe}+m\cdot p}{f_{classe}+m}$

Algorithme 6.5 – Phase d'exploitation du classifieur naïf de Bayes.

Requiert: *description*, la liste des indices générés par l'application de un ou plusieurs critères sur le vocable dont il faut déterminer la classe.

Requiert: les estimations $P(classe)$ et $P(indice/classe)$, calculées lors de la phase d'apprentissage.

Requiert: p_{min}, une constante à fournir à l'algorithme

Requiert: l'ensemble A des indices rencontrés durant la phase d'apprentissage

1: $d \leftarrow description \cap A$
2: **Si** $\max_{classe} \left(P(classe) \prod_{indice \in d} P(indice/classe) \right) > p_{min}$ **Alors**
3: **Retourner** $\operatorname{argmax}_{classe} \left(P(classe) \prod_{indice \in d} P(indice/classe) \right)$

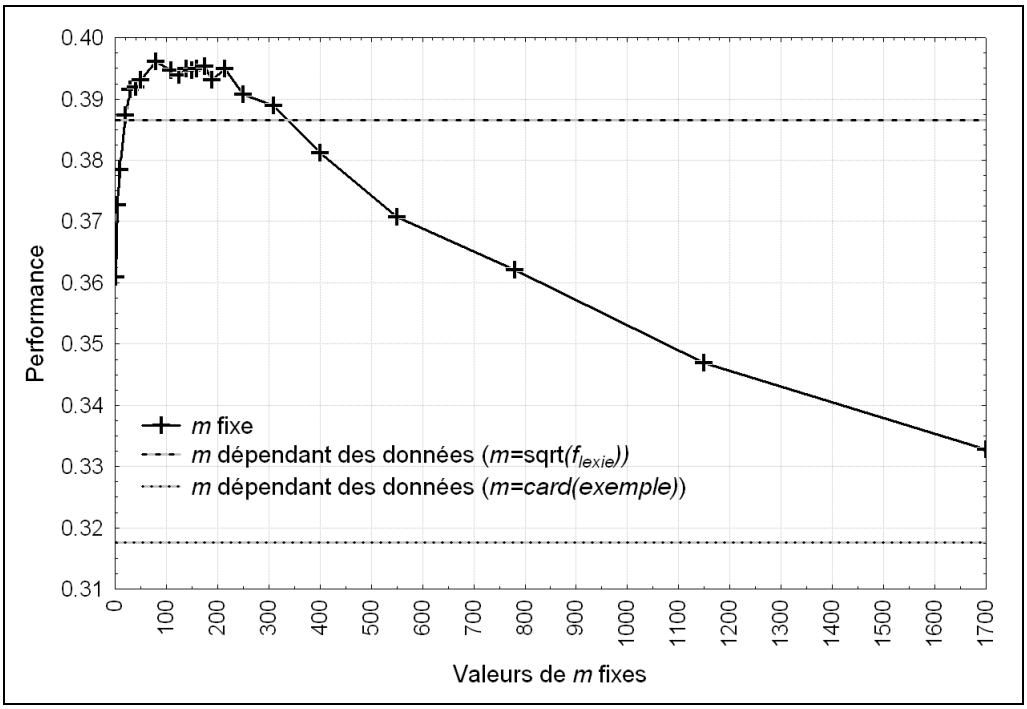


Figure 6.3 – Influence du choix de la constante m pour la m-estimation des probabilités sur les 6 vocables du sous-corpus. Pour les deux performances obtenues avec une constante m dépendante des données, nous devrions avoir deux points sur le graphique. Cependant, ces points ne peuvent être placés puisque leur position varie pour chaque classification. Nous avons donc choisi de représenter ces performances par deux droites.

	Précision	**Gain**	**Rappel**	**Performance**
Noms	74,12%	39,44%	73,94%	0,514
Adjectifs	62,04%	29,12%	61,96%	0,396
Verbes	53,32%	26,77%	53,31%	0,346
Moyenne	59,11%	25,66%	59,07%	0,383

Tableau 6.3 – Performances moyennes de l'algorithme NB(0,00) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

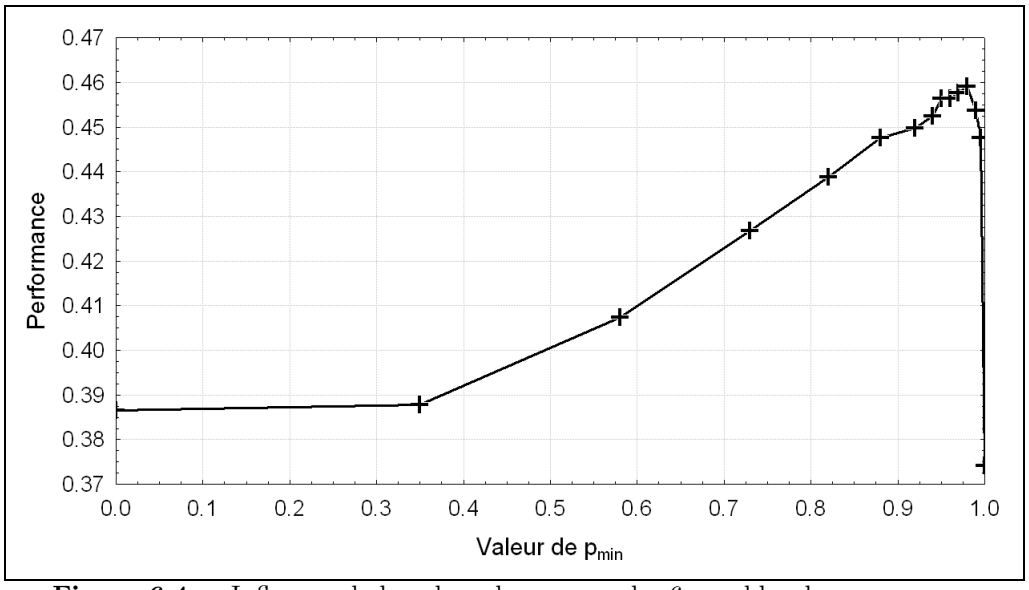


Figure 6.4 – Influence de la valeur de p_{min} sur les 6 vocables du sous-corpus.

	Précision	**Gain**	**Rappel**	**Performance**
Noms	83,06%	60,37%	63,38%	0,618
Adjectifs	75,60%	54,43%	47,11%	0,505
Verbes	66,33%	46,38%	39,55%	0,427
Moyenne	72,03%	51,00%	45,80%	0,483

Tableau 6.4 – Performances moyennes de l'algorithme NB(0,98) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

figure, où nous tenons à ce que chaque occurrence soit classée, et donc où précision et rappel sont équivalents, il est intéressant de remarquer qu'il vaut mieux utiliser l'algorithme NB(0,00), qui obtient une précision moyenne sur les 60 vocables de 59,1%, que l'algorithme NB(0,98) qui obtient lui une précision de seulement 55,6%.

Remarque

Bien que nous ayons opté pour la première représentation des attributs (cf. début de la section), la curiosité nous a poussé à savoir ce qu'il serait advenu si, conformément à Mooney (1996), nous avions choisi la deuxième représentation des attributs. Curieusement, l'algorithme équivalent à NB(0,00), mais utilisé avec la deuxième représentation des attributs, donne des résultats légèrement meilleurs que NB(0,00). Nous avons observé une augmentation du rappel et de la précision de l'ordre de 0,6%. Nous ne retenons cependant pas cet algorithme car, d'une part, il est encore plus difficile à justifier d'un point de vue théorique que NB(0,00) et, d'autre part, il est bien moins performant en terme de temps d'exécution en raison du nombre considérable d'attributs à prendre en compte (ce deuxième handicap est important compte tenu du nombre d'expériences que nous menons dans le chapitre 7).

6.5 Liste de décisions basée sur une métrique

6.5.1 Généralités

Le formalisme des listes de décision, présenté par Rivest (1987), a été utilisé avec succès par de nombreux chercheurs (Yarowsky, 1993, 1994b ; Golding, 1995 ; Mooney, 1996 ; Wilks & Stevenson, 1998 ; etc.).

Nous nous limitons ici à un certain type de listes de décision : les listes de décision basées sur une métrique. Ce type de listes de décision a été utilisé par Yarowsky (1994b), pour des mots comportant deux lexies, et généralisé par Golding (1995) aux mots à n lexies. Au contraire de l'approche bayesienne, cette approche ne combine pas l'information de tous les attributs de la description de l'occurrence dont il faut déterminer la lexie. Une liste de décisions basée sur une métrique se focalise sur un attribut unique, supposé véhiculer l'information la plus fiable d'après une métrique mesurant la *fiabilité* des attributs, pour prendre sa décision. Une telle liste de décisions peut être représentée de la manière suivante :

Si $attribut_1$ **Alors** retourner $classe_{attribut_1}$ **Sinon**
  Si $attribut_2$ **Alors** retourner $classe_{attribut_2}$ **Sinon**
    Si $attribut_3$ **Alors** retourner $classe_{attribut_3}$ **Sinon**
      $\vdots$
      Si $attribut_n$ **Alors** retourner $classe_{attribut_n}$

Les attributs $attribut_1$, $attribut_2$, ..., $attribut_n$ ont été ordonnés du plus *fiable* au moins *fiable* en utilisant la métrique. La classe $classe_{attribut_i}$ est la classe la plus probable sachant l'attribut $attribut_i$. De manière surprenante, cette stratégie semble être aussi, voire parfois plus, performante qu'une stratégie qui combine les attributs. Cette approche offre en outre un certain nombre d'avantages. Le plus important est sans doute la possibilité de traiter des attributs qui peuvent être fortement dépendants sans avoir à gérer ou à modéliser cette dépendance. Parmi les autres avantages de cette procédure, il faut mentionner la simplicité et la facilité de mise en œuvre et la transparence de

Algorithme 6.6 – Phase d'apprentissage d'une liste de décisions.

Requiert: Un ensemble d'apprentissage et une métrique mesurant la fiabilité des indices

1: Mesurer la répartition suivant les classes de chacun des indices des exemples d'apprentissage.

2: Ordonner les indices en utilisant une métrique mesurant la fiabilité des indices pour former une liste de décisions.

Algorithme 6.7 – Phase d'exploitation d'une liste de décisions.

Requiert: La liste de décisions générée durant la phase d'apprentissage

1: Parmi l'ensemble des indices de la description de l'occurrence dont il faut déterminer la classe, chercher celui qui se trouve le plus haut dans la liste de décisions.

2: Effectuer la classification en choisissant la classe la plus probable en se basant uniquement sur cet indice.

la décision à travers la liste de décisions. Par la suite, nous exploiterons régulièrement cette transparence pour mieux comprendre le mécanisme de la prise de décision.

Dans la section 6.2.2, nous avons posé le problème de la représentation des attributs. Dans une liste de décisions, les attributs doivent être binaires, nous choisissons donc la deuxième représentation dans laquelle les attributs sont les indices eux-même. La valeur d'un attribut est vrai si l'indice correspondant à cet attribut a été généré et faux s'il n'a pas été généré.

Les algorithmes 6.6 et 6.7 résument le fonctionnement global d'une méthode de classification utilisant une liste de décisions basée sur une métrique mesurant la *fiabilité* des attributs.

Il est évident que les performances d'un tel algorithme sont fortement dépendantes de la métrique choisie pour ordonner la liste de décisions. Il faut donc choisir cette métrique avec soin et il n'est pas absurde d'en essayer plusieurs comme l'a fait Golding (1995).

6.5.2 Quelques erreurs à ne pas commettre quant au choix de la métrique

Nous présentons dans cette section deux métriques qui, de prime abord, semblent pertinentes mais ne résistent pas à une étude plus approfondie. Ces quelques erreurs à ne pas commettre sont quelques erreurs que nous avons naïvement commises...

Prenons, par exemple, le vocable *populaire* avec ses cinq lexies (*i.e.* classes) : 1.1, 1.2, 1.3, 2, 3. Appliquons un critère basé sur les cooccurrences au corpus d'apprentissage. Nous comptabilisons la répartition suivant les lexies de chacun des indices des descriptions des exemples d'apprentissage. Supposons que la répartition suivant les lexies des indices *avoir, Chine, classe, clientèle, congrès, culture, et, être, front, le, presse, que, souveraineté, sport*, soit donnée par le tableau 6.5.

Dans ce tableau, il apparaît clairement que l'indice le plus fiable est *culture*, puis vient l'indice *presse*, puis *front*, *congrès*, *Chine*, etc. Nous devons donc trouver une métrique qui, suivant la fréquence des indices en fonction des lexies, nous permette d'ordonner notre liste dans l'ordre que nous venons de préciser.

L'entropie est une mesure du désordre d'un système : plus l'entropie est grande, plus grand est le désordre. Cette mesure semble tout à fait convenir à notre problème. La

indice	**1.1**	**1.2**	**1.3**	**2**	**3**
avoir	11	3	3	4	6
Chine	0	0	0	0	9
classe	0	8	0	0	0
clientèle	0	7	0	0	0
congrès	0	0	0	0	13
culture	24	0	0	0	0
et	46	12	10	8	16
être	20	5	3	4	2
front	0	0	0	0	15
le	184	40	43	43	75
presse	0	0	16	0	0
que	11	18	2	1	2
souveraineté	7	0	0	0	0
sport	0	0	0	3	0

Tableau 6.5 – Mesure de la répartition suivant les lexies de quelques indices pour le vocable *populaire*.

mesure de l'entropie $H(indice)$ des fréquences d'un indice *indice* en fonction des lexies L_1,L_2,...,L_N se calcule de la manière suivante :

$$H(indice) = -\sum_{i=1}^{i=N} p(L_i/indice).log(p(L_i/indice)) \ .$$

Une autre métrique qui semble tout à fait convenir est la mesure de dispersion. Cette mesure est une fonction de la variance comprise entre zéro et un qui traduit la dispersion des données. Plus la mesure de dispersion est proche de zéro, plus la dispersion des données est grande. Cette mesure se calcule de la manière suivante :

$$dispersion(indice) = 1 - \frac{cv(indice)}{\sqrt{N-1}}$$

où $cv(indice)$ est le coefficient de variation des fréquences de l'indice *indice* en fonction des lexies L_1,L_2,...,L_N.

Les valeurs nulles du tableau 6.5 posent des problèmes pour ces deux mesures. En remplaçant toutes ces valeurs nulles par une valeur faible (0,1), ces deux mesures ordonnent le tableau de la même manière (ce qui n'est pas toujours le cas) comme nous pouvons le voir avec le tableau 6.6. Dans la section précédente (6.5.1) nous avons parlé de la *transparence de la décision* à travers la liste de décisions, cette transparence est très bien illustrée par le tableau 6.6.

L'ordre des indices au début du tableau est bien conforme à ce que nous attendions. Nous avons testé une méthode de classification basée sur une liste de décisions utilisant l'entropie et également sur une utilisant la mesure de dispersion. Ces deux méthodes donnent de bons résultats en général. La mesure de dispersion, étant comprise entre zéro et un et possédant une dynamique plus linéaire, permet en plus d'effectuer un seuillage pour ne considérer que les indices pour lesquels la dispersion est inférieure à un certain seuil (c'est-à-dire pour écarter les indices considérés comme peu fiables). Cette méthode permet d'augmenter la précision au détriment du rappel. C'est cette méthode de classification que nous avons utilisée dans (Audibert, 2002).

indice	**1.1**	**1.2**	**1.3**	**2**	**3**	**H**	**dispersion**
culture	24	0,1	0,1	0,1	0,1	0,15	0,020
presse	0,1	0,1	16	0,1	0,1	0,21	0,030
front	0,1	0,1	0,1	0,1	15	0,23	0,032
congrès	0,1	0,1	0,1	0,1	13	0,25	0,037
Chine	0,1	0,1	0,1	0,1	9	0,34	0,053
classe	0,1	8	0,1	0,1	0,1	0,37	0,060
clientèle	0,1	7	0,1	0,1	0,1	0,41	0,068
souveraineté	7	0,1	0,1	0,1	0,1	0,41	0,068
sport	0,1	0,1	0,1	3	0,1	0,76	0,147
que	11	18	2	1	2	1,64	0,506
être	20	5	3	4	2	1,77	0,512
et	46	12	10	8	16	1,98	0,618
le	184	40	43	43	75	2,01	0,643
avoir	11	3	3	4	6	2,12	0,722

Tableau 6.6 – Liste ordonnée par entropie (**H**) ou mesure de dispersion (**dispersion**) croissante des indices du tableau 6.5.

Ces deux listes de décision se comportent correctement pour les vocables comportant deux lexies. Cependant, pour des nombres de lexies supérieurs à deux, un problème apparaît. Le tableau 6.6 illustre une matérialisation de ce problème avec l'ordonnancement des indices *que* et *être*. S'il faut classer un exemple dont l'indice le plus fiable est *que*, nous choisissons la lexie 1.2. S'il faut classer un exemple dont l'indice le plus fiable est *être*, nous choisissons la lexie 1.1. Le problème survient quand il faut classer un exemple dont les deux indices les plus fiables sont *que* et *être*. En effet, *être* est généré 20 fois sur 34 quand la lexie de *populaire* est 1.1 et *que* seulement 18 fois sur 34 quand la lexie de *populaire* est 1.2. En toute logique, il faut prendre la décision en se basant sur *être*, mais dans notre liste de décisions, *que* vient en premier car la dispersion de ses fréquences sur ses quatre lexies minoritaires est bien plus faible que pour *être*. Nous abandonnons donc ces deux listes de décision et préférons celle présentée dans la section qui suit qui ne comporte pas cette erreur d'ordonnancement et qui donne de meilleurs résultats d'après nos expériences.

6.5.3 Ordonner selon la probabilité conditionnelle maximale

Logarithme des probabilités (« log-likelihoods »)

C'est la méthode d'ordonnancement utilisée, entre autres, par Yarowsky (1993, 1994b). Pour chaque indice nous calculons le ratio :

$$\mathrm{abs}\left(\log\left(\frac{p(lexie_1/indice)}{p(lexie_2/indice)}\right)\right) \ . \tag{6.4}$$

L'indice qui désigne le plus clairement une lexie particulière a la mesure de *logarithme des probabilités* la plus grande. En classant les indices par *logarithme des probabilités* décroissant, nous les classons également par fiabilité décroissante.

Prenons par exemple le vocable *détention* avec ses deux lexies (*i.e.* classes), $L1$ pour le sens incarcération ou enfermement de détention, et $L2$ pour le sens possession de détention (cf. annexe C.4). Appliquons un critère basé sur les cooccurrences au corpus d'apprentissage. Nous comptabilisons la répartition suivant les lexies de chacun des in-

dices des descriptions des exemples d'apprentissage. Le tableau 6.7 montre la répartition suivant les lexies des indices suivants : {*être*,*provisoire*,*membre*,*condition*,*mesure*}.

indice	$L1$	$L2$
être	15	11
provisoire	10	1
membre	6	3
condition	8	2
mesure	1	2

Tableau 6.7 – Mesure de la répartition suivant les lexies de quelques indices pour le vocable *détention*.

En calculant les *logarithmes des probabilités* et en ordonnant le tableau, nous obtenons la liste de décisions :

indice i	$L1$	$L2$	$p(L1/i)$	$p(L2/i)$	$LogL(i)$
provisoire	10	1	0,91	0,09	3,3
condition	8	2	0,80	0,20	2
membre	6	3	0,67	0,33	1
mesure	1	2	0,33	0,67	1
être	15	11	0,58	0,42	0,4

Tableau 6.8 – Liste ordonnée par *logarithme des probabilités* décroissant des indices du tableau 6.7.

Prenons maintenant un exemple dont il faut déterminer la lexie la plus probable et dont la description est {*être*,*soir*,*provisoire*,*mesure*}. La classification se fait selon l'indice de *logarithme des probabilités* maximum *provisoire* qui désigne la lexie $L1$.

Probabilité conditionnelle maximale

La méthode qui vient d'être décrite est valable pour des vocables à deux lexies. La mesure suivante, proposée par Golding (1995), ordonne les attributs de la même manière que la mesure *logarithme des probabilités* (équation 6.4) mais peut être étendue à un nombre quelconque de classes :

$$fiabilité(indice) = \max_{lexies} p(lexie/indice) \ . \tag{6.5}$$

En utilisant cette mesure, l'ordre des indices du tableau 6.8 est inchangé, en revanche, l'ordre des indices du tableau 6.6 devient celui du tableau 6.9.

6.5.4 Classifieur probabilité conditionnelle maximale

Algorithmes

Soit un exemple E dont nous désirons déterminer la lexie la plus probable l_E, parmi un ensemble de lexies L, en nous basant sur la description D de E composée d'un certain nombre d'indices $D = \{i_1 \dots i_n\}$. Soit A l'ensemble des indices des exemples d'apprentissage. En appliquant la formule 6.5, nous obtenons l'indice supposé le plus fiable $IndFia$:

$$IndFia = \underset{indice \in D \cap A}{\operatorname{argmax}} \ (fiabilité(indice))$$

indice	**1.1**	**1.2**	**1.3**	**2**	**3**	***fiabilité(indice)***
culture	24	0,1	0,1	0,1	0,1	0,984
presse	0,1	0,1	16	0,1	0,1	0,976
front	0,1	0,1	0,1	0,1	15	0,974
congrès	0,1	0,1	0,1	0,1	13	0,970
Chine	0,1	0,1	0,1	0,1	9	0,957
classe	0,1	8	0,1	0,1	0,1	0,952
clientèle	0,1	7	0,1	0,1	0,1	0,946
souveraineté	7	0,1	0,1	0,1	0,1	0,946
sport	0,1	0,1	0,1	3	0,1	0,882
être	20	5	3	4	2	0,588
que	11	18	2	1	2	0,529
et	46	12	10	8	16	0,500
le	184	40	43	43	75	0,478
avoir	11	3	3	4	6	0,407

Tableau 6.9 – Liste ordonnée par *probabilité conditionnelle maximale* décroissante des indices du tableau 6.5.

avec

$$fiabilité(indice) = \max_{lexie \in L} p(lexie/indice) \ .$$

La lexie la plus probable est ensuite déterminée d'après l'indice le plus fiable :

$$l_E = \operatorname*{argmax}_{lexie \in L} p(lexie/IndFia) \ .$$

L'estimation des probabilités $p(lexie/indice)$ se fait sur les exemples d'apprentissage. Nous utilisons une m-estimation en raison de certains dénombrements faibles, et parfois nuls. En utilisant l'équation de la définition 6.14, nous obtenons :

$$P(l_l/i_i) = \frac{n_{li} + m \cdot p_{li}}{n_i + m}$$

où

- n_{li} est le nombre d'exemples d'apprentissage dont la description contient l'indice i_i et dont la lexie du vocable étudié est l_l ;
- n_i est le nombre d'exemples d'apprentissage dont la description contient un indice i_i ;
- p_{li} est une estimation *a priori* de la probabilité recherchée ; comme nous ne connaissons pas cette probabilité, nous supposons une répartition uniforme de probabilité et nous prenons $p_{li} = \frac{1}{\text{card}(L)}$ où L est l'ensemble des lexies du vocable ;
- m est une constante, appelée « équivalent de la taille de l'échantillon », à déterminer.

La phase d'apprentissage du classifieur *probabilité conditionnelle maximale* est décrite par l'algorithme 6.8.

La phase d'exploitation du classifieur *probabilité conditionnelle maximale* est décrite par l'algorithme 6.9. Dans cet algorithme, nous avons ajouté un paramètre qui permet de définir un seuil de confiance p_{min} en dessous duquel l'algorithme ne prend pas de décision.

Algorithme 6.8 – Phase d'apprentissage du classifieur *probabilité conditionnelle maximale.*

REQUIERT: Un ensemble d'apprentissage S

REQUIERT: m, une constante à fournir à l'algorithme

1: $\forall\ indice\ \forall\ lexie\ tab_{indice,lexie} \leftarrow 0$

2: $\forall\ indice\ \ fréquence_{indice} \leftarrow 0\ fiabilité_{indice} \leftarrow 0$

3: **POUR TOUT** $exemple \in S$ **FAIRE** /* Génération du tableau *tab* d'apprentissage et du vecteur $fréquence$ des fréquences des indices ; un *exemple* est une instance dans le corpus d'un vocable étudié accompagnée de sa description générée par un ou plusieurs critères et de sa *lexie* d'appartenance. */

4: **POUR TOUT** $indice \in exemple$ **FAIRE**

5: $tab_{indice,lexie} \leftarrow tab_{indice,lexie} + 1$ où *lexie* est la lexie de *exemple*

6: $fréquence_{indice} \leftarrow fréquence_{indice} + 1$

7: $p \leftarrow \frac{1}{\text{nombre de lexies}}$ /* Estimation de la probabilité *a priori* */

8: **POUR TOUT** *indice* **FAIRE** /* Mesure de la fiabilité des indices */

9: $fiabilité_{indice} \leftarrow \max_{lexie \in L} \left(\frac{tab_{lexie,indice} + m \cdot p}{fréquence_{indice} + m} \right)$

Algorithme 6.9 – Phase d'exploitation du classifieur *probabilité conditionnelle maximale.*

REQUIERT: *description*, la liste des indices générés par l'application de un ou plusieurs critères sur le vocable dont il faut déterminer la classe.

REQUIERT: les estimations $fiabilité_{indice}$, calculées lors de la phase d'apprentissage

REQUIERT: le tableau $tab_{indice,classe}$ de fréquence des indices en fonction des classes généré lors de la phase d'apprentissage

REQUIERT: p_{min}, une constante à fournir à l'algorithme

REQUIERT: l'ensemble A des indices rencontrés durant la phase d'apprentissage

1: $D \leftarrow description \cap A$

2: $IndFia = \underset{indice \in D}{\text{argmax}}(fiabilité(indice))$ /* Sélection de l'indice le plus fiable */

3: **SI** $IndFia \neq \phi$ et $fiabilité(indice) > p_{min}$ **ALORS**

4: **RETOURNER** $\underset{classe}{\text{argmax}}\,(tab_{classe,IndFia})$

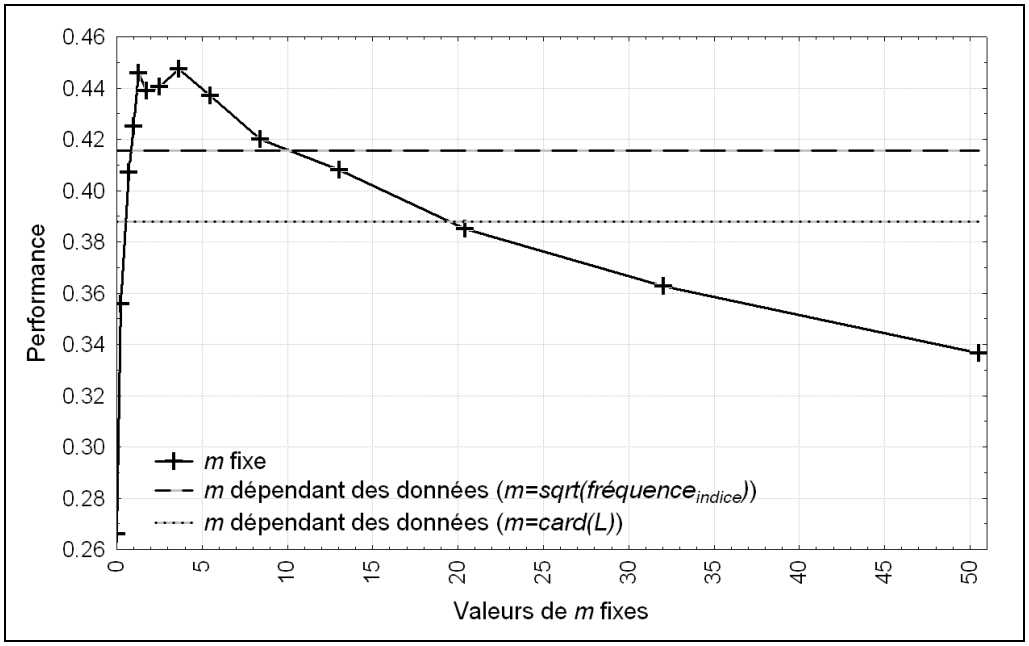


Figure 6.5 – Influence du choix de la constante m pour la m-estimation des probabilités sur les 6 vocables du sous-corpus. Pour les deux performances obtenues avec une constante m dépendante des données, nous devrions avoir deux points sur le graphique. Cependant, ces points ne peuvent être placés puisque leur position varie pour chaque classification. Nous avons donc choisi de représenter ces performances par deux droites.

Affinage et performances

Affiner cet algorithme consiste à déterminer la meilleure constante m. Nous avons choisi d'essayer plusieurs valeurs fixes de m et deux valeurs dépendant des données $m = \sqrt{fréquence_{indice}}$ et $m = \text{card}(L)$.

Dans le cas où $m = 0$, nous retrouvons une estimation de probabilité classique :

$$P(l_l/i_i) = \frac{n_{li} + m \cdot p_{li}}{n_i + m} = \frac{n_{li}}{n_i}$$

Dans le cas où $m = \text{card}(L)$, nous avons :

$$P(l_l/i_i) = \frac{n_{li} + m \cdot p_{li}}{n_i + m} = \frac{n_{li} + \text{card}(L) \cdot \dfrac{1}{\text{card}(L)}}{n_i + \text{card}(L)} = \frac{n_{li} + 1}{n_i + \text{card}(L)}$$

Ce cas de figure correspond à la façon d'estimer les probabilités dans l'algorithme de classification de l'étude préliminaire. Cela revient à faire un lissage de Laplace, utilisé entre autres par Golding (1995).

La figure 6.5 montre les performances obtenues par l'algorithme en fonction de différentes valeurs de m sur les 6 vocables du sous-corpus. Dans ces expériences $p_{min} = 0$, l'algorithme classifie donc le maximum d'exemples possible. La valeur de m permettant d'obtenir les meilleures performances se situe entre 1 et 4. Nous remarquons que le lissage réalisé par Golding, avec $m = \text{card}(L)$, donne de moins bons résultats que la valeur de $m = \sqrt{fréquence_{indice}}$. Comme nous l'avons dit pour le classifieur naïf de

Bayes (section 6.4.4), avec d'autres critères ou d'autres vocables, il est probable que la valeur de m fixe permettant d'obtenir les meilleures performances ne soit plus la même. Choisir une valeur de m dépendante des données, plutôt qu'une constante déterminée expérimentalement, nous paraît donc une meilleure solution.

Nous nommons PCM(0,00) l'algorithme 6.8 avec $m = \sqrt{fréquence_{indice}}$ et l'algorithme 6.9 avec $p_{min} = 0$. Le tableau 6.10 résume les performances moyennes de l'algorithme PCM(0,00) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

Si nous ne cherchons pas à maximiser le rappel, il est possible d'améliorer la performance, c'est-à-dire la moyenne harmonique du rappel et du gain, en augmentant le seuil de confiance p_{min}. La figure 6.6 montre l'influence de la valeur du seuil p_{min} sur la performance moyenne pour les 6 vocables. La valeur du seuil p_{min} optimale semble se situer autour de 0,575. Dès que nous dépassons cette valeur, la chute rapide du rappel entraîne une chute brutale des performances.

Une valeur du seuil de confiance $p_{min} = 0{,}57$ nous semble être un bon compromis. Nous désignons par PCM(0,57) l'algorithme 6.8 avec $m = \sqrt{fréquence_{indice}}$ et l'algorithme 6.9 avec $p_{min} = 0{,}57$. Le tableau 6.11 résume les performances moyennes de l'algorithme PCM(0,57) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

6.6 Apprentissage basé sur les instances

6.6.1 Aspect théorique

Généralités

Contrairement à des méthodes qui construisent explicitement une procédure de classification lors de l'apprentissage, les méthodes d'apprentissage basé sur les instances se contentent de mémoriser chacun des exemples. Il ne s'agit donc pas de méthodes inductives. L'exploitation des exemples d'apprentissage pour la prise de décision est repoussée à la phase d'exploitation, c'est-à-dire au moment de la rencontre du nouvel exemple. À chaque nouvel exemple à classer, une exploration des relations entre cet exemple et les exemples mémorisés est réalisée dans le but de déterminer la classe de l'exemple à classer.

L'inconvénient majeur de ce type de classification est que la complexité de la classification de chaque nouvel exemple peut être élevée. En effet, la majeure partie des calculs est effectuée au moment de chaque classification plutôt qu'au moment de l'apprentissage comme c'est le cas pour les autres méthodes présentées.

Méthodes du type k plus proches voisins

La méthode d'apprentissage basé sur les instances la plus classique est l'algorithme des k plus proches voisins. De nombreuses études (Cost & Salzberg, 1993 ; Escudero *et al.*, 2000c ; Mooney, 1996 ; Ng, 1997 ; etc.) montrent que cette méthode de classification peut rivaliser, voire surpasser, le classifieur naïf de Bayes.

La méthode des k plus proches voisins associe à chaque exemple un point dans un espace à n dimensions. Lors de la phase d'apprentissage, toutes les descriptions des exemples sont mémorisées. Pendant la phase d'exploitation, lorsqu'un exemple est rencontré, il faut calculer la distance entre la description de cet exemple et la description de tous les exemples mémorisés pendant la phase d'apprentissage. L'algorithme sélectionne

	Précision	Gain	Rappel	Performance
Noms	75,79%	43,35%	75,61%	0,551
Adjectifs	64,29%	33,32%	64,21%	0,439
Verbes	57,00%	31,52%	56,99%	0,406
Moyenne	62,13%	33,65%	62,09%	0,436

Tableau 6.10 – Performances moyennes de l'algorithme PCM(0,00) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

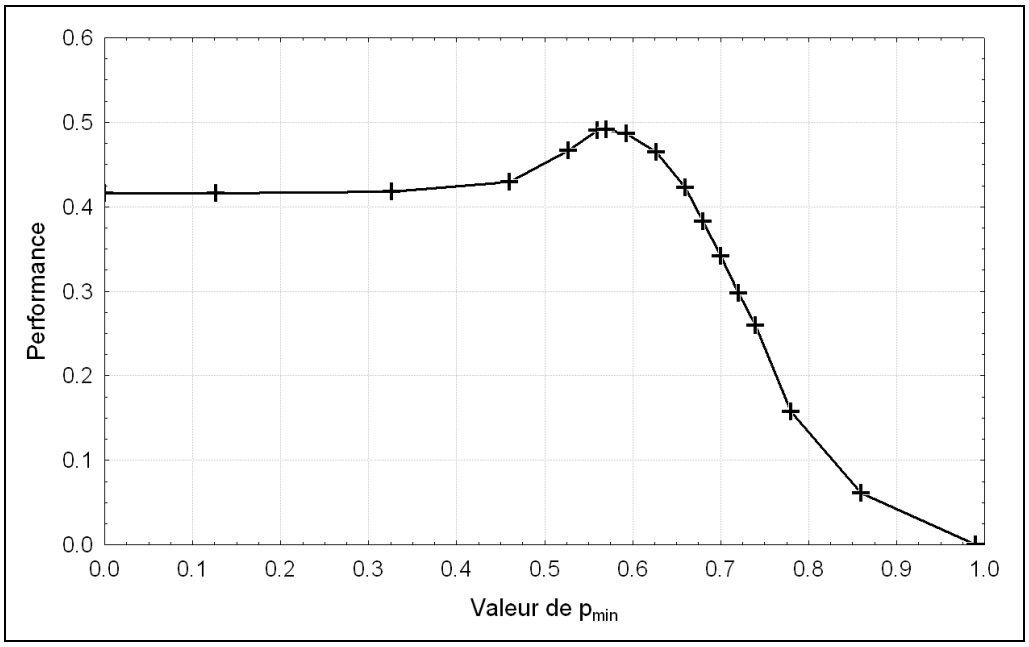


Figure 6.6 – Influence de la valeur du seuil de confiance sur les 6 vocables du sous-corpus.

	Précision	Gain	Rappel	Performance
Noms	80,26%	53,81%	70,18%	0,609
Adjectifs	75,76%	54,74%	55,05%	0,549
Verbes	73,03%	57,05%	44,13%	0,498
Moyenne	75,46%	57,00%	51,39%	0,541

Tableau 6.11 – Performances moyennes de l'algorithme PCM(0,57) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

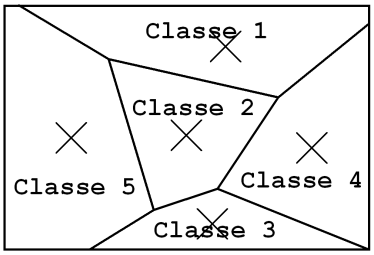


Figure 6.7 – Exemple de diagramme de Voronoï (cinq classes et deux attributs).

Algorithme 6.10 – Phase d'apprentissage générale d'une méthode du type k plus proches voisins.

1: Mémoriser tous les exemples *exemple* d'apprentissage.

ensuite les k exemples les plus proches et décide que la classe de l'exemple rencontré est la classe majoritairement représentée dans les k exemples.

Dans le cas où $k = 1$, nous pouvons caractériser précisément les zones de décision en nous ramenant à un diagramme de Voronoï tel que celui présenté figure 6.7.

Les algorithmes 6.10 et 6.11 résument le fonctionnement global d'une méthode du type k plus proches voisins.

Le dernier problème qu'il reste à régler est la définition de la distance d. Malheureusement, il n'existe pas de définition universelle de cette distance d.

Distance Euclidienne

Si tous les attributs sont numériques et homogènes (*i.e.* normalisés), nous pouvons utiliser une distance Euclidienne simple. Notons x un exemple de description $\{x_1,\ldots,x_n\}$ et y un exemple de description $\{y_1,\ldots,y_n\}$ où x_i et y_i font référence aux valeurs du même attribut i. Alors, la distance $D_E(x,y)$ entre les exemples x et y est :

$$D_E(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Distance de Hamming

Dans le cas où les attributs sont nominaux, nous pouvons utiliser la distance de Hamming. Notons x un exemple de description $\{x_1,\ldots,x_n\}$ et y un exemple de description $\{y_1,\ldots,y_n\}$ où x_i et y_i font référence aux valeurs du même attribut i. Alors, la distance de Hamming $D_H(x,y)$ entre les exemples x et y est :

$$D_H(x,y) = \sum_{i=1}^{n}\delta(x_i,y_i)$$

où $\delta(x_i,y_i) = 1$ si $x_i = y_i$ et $\delta(x_i,y_i) = 0$ sinon.

Mesure d'information positive (similarité)

La distance de Hamming entre deux descriptions peut être calculée en se plaçant dans le cadre de la seconde représentation des attributs présentée section 6.2.2, lorsque

Algorithme 6.11 – Phase d'exploitation générale d'une méthode du type k plus proches voisins.

REQUIERT: tous les exemples d'apprentissage

1: Rechercher, parmi les exemples mémorisés durant la phase d'apprentissage, l'ensemble $E_k = \{exemple_1, \ldots, exemple_k\}$ des k exemples les plus proches, selon une mesure de distance d donnée, de l'exemple dont il faut déterminer la classe.

2: Effectuer la classification en choisissant la classe la plus fréquente de l'ensemble E_k.

nous avons posé le problème de la représentation des attributs. En nous plaçant dans le cadre de la première représentation, nous pouvons mesurer l'information positive (habituellement appelée *matching coefficient* en anglais), ou similarité S (Escudero *et al.*, 2000c), entre deux exemples x et y de descriptions respectives $D_x = \{x_1, \ldots, x_n\}$ et $D_y = \{y_1, \ldots, y_n\}$ de la manière suivante :

$$S(x,y) = card(D_x \cap D_y) \tag{6.6}$$

Distance *modified value difference metric* (MVDM)

Cette distance a été proposée par Cost et Salzberg (1993) et notamment utilisée par Escudero, Marquez et Rigau (2000c), Ng (1997). Il s'agit d'une adaptation de la distance *value difference metric* VDM proposée par Stanfill et Waltz (1986). Plaçons nous dans un problème à m classes $C_1, \ldots, C_m$. La distance MVDM entre deux valeurs symboliques v_1 et v_2 se calcule de la manière suivante :

$$d_{MVDM}(v_1,v_2) = \sum_{i=1}^{m} |P(C_i/v_1) - P(C_i/v_2)| \tag{6.7}$$

Ce qui peut être estimé de la manière suivante :

$$d_{MVDM}(v_1,v_2) \approx \sum_{i=1}^{m} \left| \frac{N_{v_1,i}}{N_{v_1}} - \frac{N_{v_2,i}}{N_{v_2}} \right|$$

où $N_{v_x,i}$ est le nombre d'exemples d'apprentissage dont la valeur de l'attribut a est v_x et dont la classe est i, et N_{v_x} le nombre d'exemples d'apprentissage dont la valeur de l'attribut a est v_x toutes classes confondues.

Nous pouvons maintenant calculer la distance $D_{MVDM}(x,y)$ entre deux exemples x et y de descriptions respectives $\{x_1, \ldots, x_n\}$ et $\{y_1, \ldots, y_n\}$ de la manière suivante :

$$D_{MVDM}(x,y) = w_x.w_y. \sum_{i=1}^{n} (f_i.d_{MVDM}(x_i,y_i)^r) \tag{6.8}$$

L'algorithme de classification utilisant cette distance est appelé PEBLS (Cost & Salzberg, 1993). w_x et w_y sont des termes qui pondèrent l'importance accordée aux exemples x et y. Le terme f_i n'a été introduit qu'à partir de la version 3.0 de l'implémentation de PEBLS. Il permet de pondérer l'importance accordée à l'attribut i. En choisissant $r = 1$ nous obtenons une distance de Manhattan et en choisissant $r = 2$ une distance Euclidienne.

Algorithme 6.12 – Phase d'apprentissage du classifieur k plus proches voisins.

REQUIERT: $A = \{exemple_1, \ldots, exemple_n\}$ est l'ensemble des exemples d'apprentissage /* Un *exemple* est une instance dans le corpus d'un vocable étudié accompagnée de sa description générée par un ou plusieurs critères et de sa *classe* d'appartenance. */

1: **POUR TOUT** $exemple_i$ **FAIRE** /* Constitution du tableau *tab* mémorisant tous les exemples d'apprentissage */

2: $tab[i] \longleftarrow exemple_i$.

6.6.2 Classifieur k plus proches voisins

Choix des attributs et de la mesure de distance

Dans la section 6.2.2, nous avons posé le problème de la représentation des attributs. Escudero, Marquez et Rigau (2000c) ont montré que le choix de la première représentation est intéressant du point de vue de la complexité (réduction du temps de calcul) et des performances. Pour cet algorithme, nous utilisons donc la formule 6.6 qui mesure l'information positive. Plus deux exemples sont proches (similaires), plus cette mesure est grande. Pour obtenir une distance entre deux exemples x et y de descriptions respectives $D_x = \{x_1, \ldots, x_n\}$ et $D_y = \{y_1, \ldots, y_n\}$ qui décroît avec la similarité des exemples x et y, nous prenons l'inverse de cette mesure plus un :

$$D_S(x,y) = \frac{1}{S(x,y)+1} = \frac{1}{card(D_x \cap D_y)+1} \tag{6.9}$$

Bien entendu, nous ajoutons un à $S(x,y)$ pour le cas où $S(x,y) = 0$.

Algorithmes

La phase d'apprentissage du classifieur k plus proches voisins est décrite par l'algorithme 6.12.

La phase d'exploitation du classifieur k plus proches voisins est décrite par l'algorithme 6.13. Quand plusieurs lexies sont en compétition, parce qu'elles sont représentées de manière égale dans l'ensemble des k plus proches voisins, nous choisissons celle qui est la plus fréquente dans le corpus d'apprentissage.

Ces deux algorithmes sont des versions plus détaillées des algorithmes 6.10 et 6.11 qui expliquent simplement la méthode des k plus proches voisins.

Affinage et performances

Affiner cet algorithme consiste à déterminer la meilleure constante k.

La figure 6.8 montre la performance moyenne obtenue pour les 6 vocables pour des valeurs de k comprises entre 1 et 100.

La meilleure valeur de k semble se situer autour de 20. Nous désignons par KPPV(20) l'algorithme 6.13 avec une valeur de $k = 20$.

Le tableau 6.12 résume les performances moyennes pour chacune des trois catégories de vocables de l'algorithme KPPV(20). Cet algorithme prenant une décision sur toutes les instances à étiqueter, nous avons Précision = Rappel. Pour cette raison, nous n'avons pas fait figurer le rappel dans le tableau. Nous pouvons déjà remarquer que les performances de cet algorithme sont très en retrait par rapport à celles des algorithmes précédents.

Algorithme 6.13 – Phase d'exploitation du classifieur k plus proches voisins.

REQUIERT: *description*, la liste des indices générés par l'application de un ou plusieurs critères sur le vocable dont il faut déterminer la lexie.

REQUIERT: *tab*, le tableau, généré lors de l'apprentissage, mémorisant tous les exemples d'apprentissage.

REQUIERT: D_S, la mesure de distance définie par l'équation 6.9

REQUIERT: k, la constante indiquant le nombre de voisins à prendre en compte pour effectuer une classification.

1: soit σ une permutation sur $\{1, \ldots, card(tab)\}$ telle que $\sigma(i) < \sigma(j)$ si et seulement si $D_S(description,tab[i]) < D_S(description,tab[j])$ /* Ordonnancement des exemples d'apprentissage en fonction de leur distance à la description de l'occurrence dont il faut déterminer la classe */

2: soit $voisins = \{exemple_i | \sigma(i) < k\}$ /* *voisin* est l'ensemble des k plus proches voisins de la description de l'occurrence dont il faut déterminer la classe */

3: **RETOURNER** $\underset{classe}{\text{argmax}} \left(\sum_{exemple \in voisins} \delta(classe,exemple) \right)$ avec $\delta(l,e) = 1$ si la classe de l'exemple e est l et $\delta(l,e) = 0$ sinon /* Retourne la classe la plus représentée dans l'ensemble *voisin*. **Remarque** : en cas de conflit entre plusieurs classes, l'algorithme retourne celle qui est la plus fréquente dans le corpus d'apprentissage */

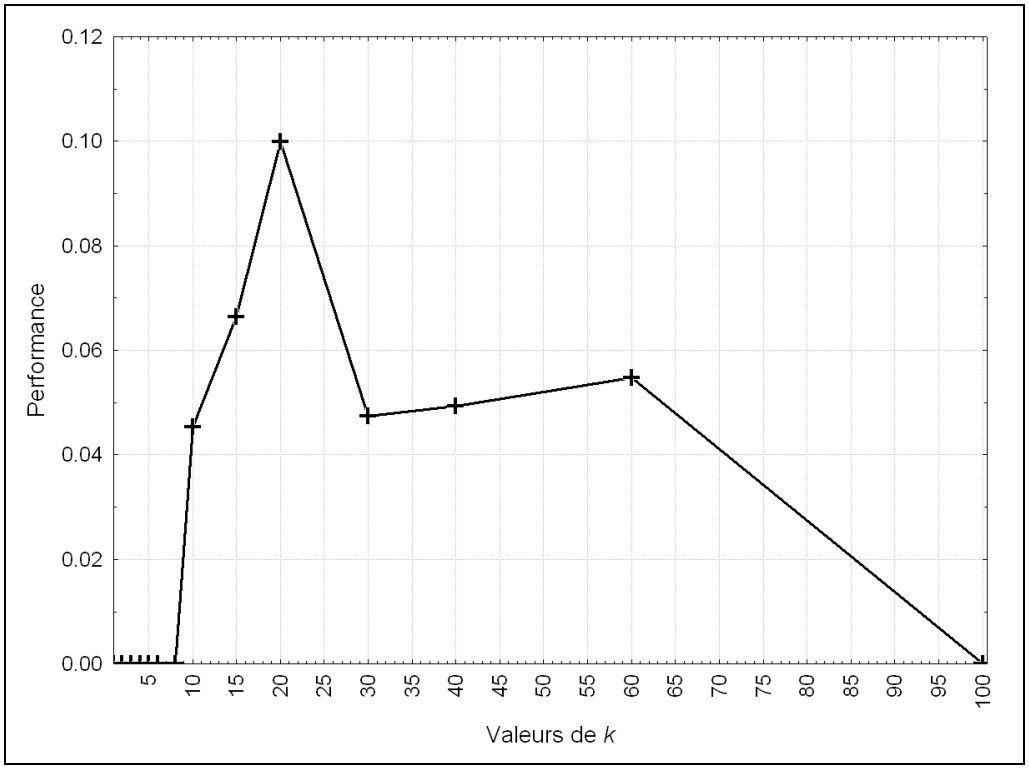


Figure 6.8 – Performances moyennes du classifieur k plus proches voisins sur les 6 vocables du sous-corpus.

	Précision	Gain	Performance
Noms	64,91%	17,92%	0,281
Adjectifs	51,24%	8,96%	0,153
Verbes	39,24%	3,25%	0,060
Moyenne	46,60%	6,44%	0,113

Tableau 6.12 – Performances moyennes de l'algorithme KPPV(20) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

6.6.3 Classifieur PEBLS (*Parallel Exemplar-Based Learning System*)

Choix des attributs et de la mesure de distance

Dans la section 6.2.2, nous avons posé le problème de la représentation des attributs. Escudero, Marquez et Rigau (2000c) ont montré que le choix de la première représentation est intéressant du point de vue de la complexité (réduction du temps de calcul) et des performances. Pour cet algorithme, la mesure de distance est donnée par les formules 6.7 et 6.8. En raison de notre choix de la représentation des attributs, il faut légèrement adapter la façon d'estimer les probabilités de la formule 6.7. Nous les estimons de la manière suivante :

$$d_{MVDM}(v_1,v_2) \approx \sum_{i=1}^{m} \left| \frac{N_{v_1,i}}{N_{v_1}} - \frac{N_{v_2,i}}{N_{v_2}} \right|$$

L'adaptation se fait au niveau de la façon de calculer les valeurs de $N_{v_x,i}$ et N_{v_x} :

- $N_{v_x,i}$ est le nombre d'exemples d'apprentissage dont la description contient un attribut de valeur v_x et dont la classe est i ;
- N_{v_x} le nombre d'exemples d'apprentissage dont la description contient un attribut de valeur v_x toutes classes confondues.

Conformément à la formule 6.8, la mesure finale entre deux exemples s'obtient de la manière suivante :

$$D_{MVDM}(x,y) = w_x.w_y. \sum_{i=1}^{n} (f_i.d_{MVDM}(x_i,y_i)^r)$$

Dans notre expérience, nous n'utilisons pas les termes qui pondèrent l'importance accordée aux exemples ou aux attributs : nous posons $w_x = w_y = f_i = 1$. Conformément à ce que préconisent Cost et Salzberg, nous choisissons également $r = 1$. Notre distance devient donc :

$$D_{MVDM}(x,y) = \sum_{i=1}^{n} d_{MVDM}(x_i,y_i) \tag{6.10}$$

Algorithmes

L'algorithme est celui des k plus proches voisins en utilisant la mesure de distance précisée ci-dessus.

La phase d'apprentissage du classifieur PEBLS est décrite par l'algorithme 6.14.

La phase d'exploitation du classifieur PEBLS est décrite par l'algorithme 6.15. Quand plusieurs lexies sont en compétition, parce qu'elles sont représentées de manière égale

Algorithme 6.14 – Phase d'apprentissage du classifieur PEBLS.

REQUIERT: $A = \{exemple_1, \ldots, exemple_n\}$ est l'ensemble des exemples d'apprentissage /* Un *exemple* est une instance dans le corpus d'un vocable étudié accompagnée de sa description générée par un ou plusieurs critères et de sa *classe* d'appartenance. */

1: **POUR TOUT** $exemple_i$ **FAIRE** /* Constitution du tableau *tab* mémorisant tous les exemples d'apprentissage */

2: $tab[i] \longleftarrow exemple_i$.

Algorithme 6.15 – Phase d'exploitation du classifieur PEBLS.

REQUIERT: *description*, la liste des indices générés par l'application de un ou plusieurs critères sur l'occurrence dont il faut déterminer la classe.

REQUIERT: *tab*, le tableau, généré lors de l'apprentissage, mémorisant tous les exemples d'apprentissage.

REQUIERT: D_S, la mesure de distance définie par l'équation 6.10

REQUIERT: k, la constante indiquant le nombre de voisins à prendre en compte pour effectuer une classification.

1: soit σ une permutation sur $\{1, \ldots, card(tab)\}$ telle que $\sigma(i) < \sigma(j)$ si et seulement si $D_S(description,tab[i]) < D_S(description,tab[j])$ /* Ordonnancement des exemples d'apprentissage en fonction de leur distance à la description de l'occurrence dont il faut déterminer la classe */

2: soit $voisins = \{exemple_i | \sigma(i) < k\}$ /* *voisin* est l'ensemble des k plus proches voisins de la description de l'occurrence dont il faut déterminer la classe */

3: **RETOURNER** $\underset{classe}{\text{argmax}} \left(\sum_{exemple \in voisins} \delta(classe,exemple) \right)$ avec $\delta(l,e) = 1$ si la classe de l'exemple e est l et $\delta(l,e) = 0$ sinon /* Retourne la classe la plus représentée dans l'ensemble *voisin*. **Remarque** : en cas de conflit entre plusieurs classes, le classifieur retournera celle qui est la plus fréquente dans le corpus d'apprentissage */

dans l'ensemble des k plus proches voisins, nous choisissons celle qui est la plus fréquente dans le corpus d'apprentissage.

Ces deux algorithmes sont des versions plus détaillées des algorithmes 6.10 et 6.11 qui expliquent simplement la méthode des k plus proches voisins. Mise à part la définition de la distance, ces algorithmes sont les mêmes que ceux du classifieur k plus proches voisins, section 6.6.2.

Affinage et performances

Affiner cet algorithme consiste à déterminer la meilleure constante k.

La figure 6.9 montre la performance moyenne obtenue pour chacune des catégories de vocables et pour les 6 vocables pour des valeurs de k comprises entre 1 et 100.

La meilleure valeur de k se situe autour de $k = 5$ ($performance = 0{,}406$). Nous désignons par PEBLS(5) l'algorithme 6.15 avec une valeur de $k = 5$.

Le tableau 6.13 résume les performances moyennes de l'algorithme PEBLS(5) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables. Cet algorithme prenant une décision sur toutes les instances à étiqueter, nous avons Précision = Rappel. Pour cette raison, nous n'avons pas fait figurer le rappel dans le tableau.

6.7 Synthèse des résultats des classifieurs évalués

Comme nous l'avons précisé dans la section 6.3.2, nous avons utilisé un sous-corpus pour l'affinage et l'évaluation des classifieurs. Il serait risqué de comparer les performances des classifieurs sur ce même sous-corpus puisqu'il a été utilisé pour affiner les paramètres des classifieurs. Nous choisissons donc de comparer les performances des classifieurs sur un sous-corpus distinct réalisé autour de six autres vocables (deux noms, deux adjectifs et deux verbes) : *constitution*, *économie*, *sensible*, *sûr*, *poursuivre* et *rendre*.

Le tableau 6.14 résume les performances moyennes des sept classifieurs évalués pour ces 6 vocables. La durée, exprimée en secondes, est le temps mis par le classifieur pour effectuer l'apprentissage et la classification des 6 vocables en utilisant la méthode de validation croisée k-fois.

Comme nous l'avons déjà remarqué, le classifieur KPPV(20) ne donne pas de bons résultats par rapport aux autres. D'autre part, il est relativement lent : 93 secondes de traitement à comparer aux 2 secondes du classifieur le plus rapide. Nous ne l'utiliserons plus par la suite.

Les performances du classifieur PEBLS(5) sont bonnes, elles surpassent même celles du classifieur naïf de Bayes (NB(0,00)). Hélas, son temps d'exécution est prohibitif : il met 663 secondes (soit 11 minutes et 3 secondes) là où le classifieur naïf de Bayes (NB(0,00)) et la liste de décisions (PCM(0,00)) en mettent respectivement 3 et 2. Nous ne pourrons, en raison de son temps de traitement, utiliser systématiquement ce classifieur par la suite.

Le classifieur NB(0,00) est très rapide en temps d'exécution, il se rapproche du classifieur PCM(0,00) et se détache franchement des classifieurs du type k plus proches voisins (KPPV(20) et PEBLS(5)). Ce résultat va à l'encontre de ceux présentés par Mooney (1996) qui donne le classifieur naïf de Bayes comme étant le plus lent de son étude qui comporte pourtant un classifieur du type k plus proches voisins. La raison est double : d'une part, Mooney utilise dans son expérience la deuxième représentation des attributs plutôt que la première (cf. section 6.2.2), d'autre part, Mooney n'effectue

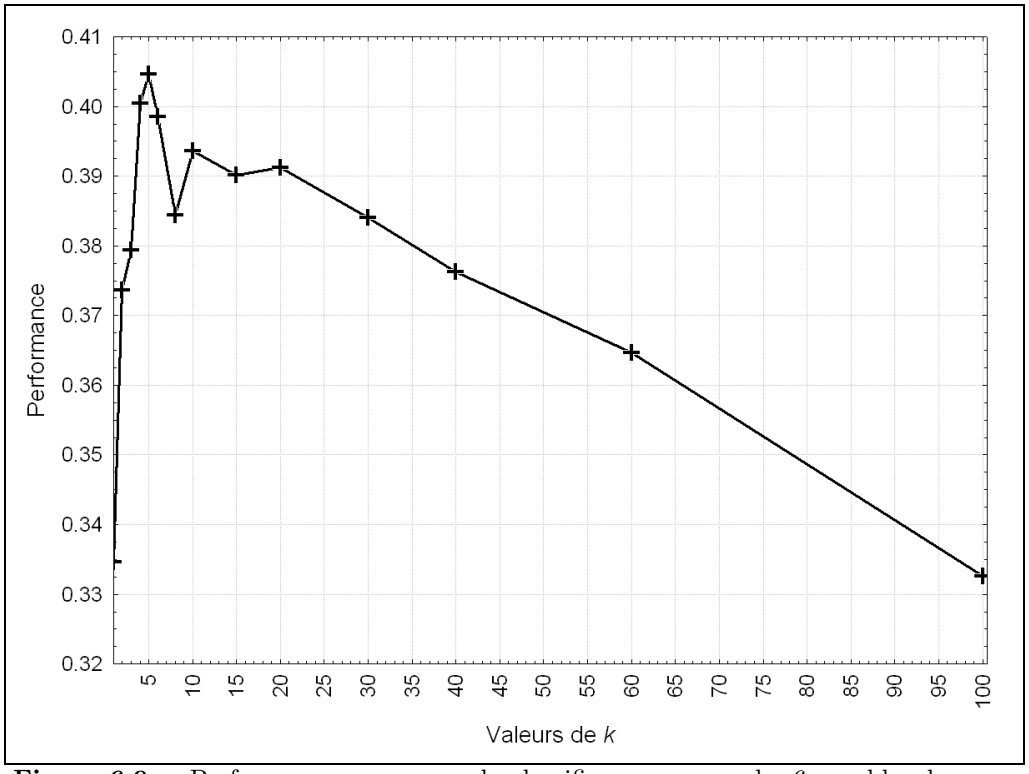


Figure 6.9 – Performances moyennes du classifieur PEBLS sur les 6 vocables du sous-corpus.

	Précision	**Gain**	**Performance**
Noms	74,71%	40,82%	0,528
Adjectifs	62,41%	29,82%	0,404
Verbes	54,52%	27,58%	0,366
Moyenne	60,06%	30,01%	0,400

Tableau 6.13 – Performances moyennes de l'algorithme PEBLS(5) pour chacune des trois catégories grammaticales et pour l'ensemble des 60 vocables.

	MAJ	NB(0,00)	NB(0,98)	PCM(0,00)	PCM(0,57)	KPPV(20)	PEBLS(5)
Précision	43,9%	59,8%	71,9%	65,8%	75,1%	51,8%	64,6%
Rappel	43,9%	59,8%	45,6%	65,8%	56,5%	51,8%	64,6%
Gain	0,0%	28,3%	49,8%	39,1%	55,6%	14,0%	36,9%
Performance	0,00	0,38	0,48	0,49	0,56	0,22	0,47
Durée (en s.)	1	3	3	2	2	93	663

Tableau 6.14 – Synthèse des résultats des classifieurs évalués sur les vocables *constitution*, *économie*, *sensible*, *sûr*, *poursuivre* et *rendre*.

pas l'estimation des probabilités, qui peut être réalisée une fois pour toutes, lors de la phase d'apprentissage, mais effectue ce calcul pour chaque instance à classer lors de la phase d'exploitation !

Chapitre 7

Étude de critères pour la désambiguïsation lexicale automatique

7.1 Introduction

La désambiguïsation lexicale s'effectue toujours en utilisant l'information du contexte du mot à désambiguïser. Cette information peut être enrichie par un certain nombre d'annotations (étiquette morphosyntaxique, lemmatisation, etc.). Cette information peut également être utilisée conjointement avec des bases de connaissances externes. Dans tous les cas, il n'est pas possible d'utiliser toute l'information disponible car elle est bien trop bruitée. Il faut donc se focaliser sur un certain nombre d'indices. Le choix de ces indices, déterminé par ce que nous appelons des critères de désambiguïsation lexicale, est primordial et constitue un enjeu important en désambiguïsation lexicale (Bruce *et al.*, 1996 ; Ng & Zelle, 1997). En effet, la précision de la désambiguïsation dépend en grande partie de ce choix et dans une moindre partie du choix de l'algorithme de classification (Pedersen, 2001).

L'objectif de ce chapitre est de réaliser une étude systématique et approfondie de critères pour la désambiguïsation lexicale automatique. Les ressources dont nous disposons nous permettent d'étudier des critères basés sur des cooccurrences de mots, et plus généralement de n-grammes (juxtaposition de un ou plusieurs mots). Nous tentons de répondre aux multiples questions que l'utilisation de tels critères soulève, comme la taille et la symétrie des contextes à considérer, l'importance de la lemmatisation, de l'ordre des mots, des mots grammaticaux, de la taille des n-gramme utilisés, etc.

Avant de mener à bien cette étude, nous nous intéressons aux travaux proches du nôtre effectués par d'autres équipes et nous exposons notre protocole expérimental de manière à positionner correctement notre étude (**section 7.2**).

Nous entreprenons ensuite une étude systématique et approfondie de critères isolés basés sur les cooccurrences (**section 7.3**) puis sur les n-grammes avec $n > 1$ en nous intéressant particulièrement aux bigrammes et aux trigrammes (**section 7.4**).

Dans la **section 7.5**, nous tentons de combiner les critères étudiés dans les sections précédentes, de manière à améliorer la précision de la désambiguïsation.

La **section 7.6** synthétise les résultats obtenus.

Nous clôturons ce chapitre en discutant des mesures prises tout au long de ce travail

de thèse afin de réduire au maximum les éventuelles erreurs ou biais pouvant entacher les résultats que nous présentons (**section 7.7**).

7.2 Positionnement de notre étude

7.2.1 Critères étudiés par d'autres équipes

Avant de mener à bien notre étude sur les critères de désambiguïsation lexicale automatique, il est intéressant de regarder les critères utilisés par d'autres équipes dans le cadre d'études similaires à la nôtre. Nous dressons ici une liste, non exhaustive, de critères tirant l'information de corpus annotés pour alimenter des algorithmes de classification supervisée. Nous faisons suivre l'énoncé des critères par une liste des travaux les utilisant. Nous ne citons pas les travaux qui restent trop vagues sur les critères utilisés, ou qui renvoient directement aux critères utilisés par d'autres travaux. Nous avons choisi d'énoncer les critères de manière très générale ; nous précisons, quand c'est possible, la façon dont ces critères sont interprétés dans chacune des études citées. Cette précision est souvent incomplète : Yarowsky (2000), par exemple, évoque un grand nombre de combinaisons possibles pour former des critères paramétrables en ne précisant ni la valeur des paramètres ni si toutes les combinaisons sont utilisées.

Cooccurrences

Dans le cadre de notre étude, nous définissons le terme *cooccurrence* de la manière suivante :

Définition 7.1 – Cooccurrence –
Nous désignons par le terme générique de cooccurrence d'un mot x un ou plusieurs mots apparaissant dans le contexte du mot x sans contrainte de figement ou de lien syntaxique.

Les cooccurrences peuvent être différenciées en fonction de leur position relative au mot à désambiguïser. Elles peuvent également revêtir la forme du jeton (forme fléchie), du lemme ou de l'étiquette morphosyntaxique du mot qu'elle désigne. Voici une liste non exhaustive de critères basés sur les cooccurrences utilisés par d'autres équipes dans le cadre d'études similaires à la nôtre :

1. « 50 **mots à gauche et à droite du mot à désambiguïser** » (Gale, Church & Yarowsky, 1992b)
2. « **mot plein adjacent et situé à droite du mot à désambiguïser** » (Yarowsky, 1993)
3. « **mot plein adjacent et situé à gauche du mot à désambiguïser** » (Yarowsky, 1993)
4. « **premier mot plein situé à droite du mot à désambiguïser** » (Yarowsky, 1993)
5. « **premier mot plein situé à gauche du mot à désambiguïser** » (Yarowsky, 1993)
6. « **mots situés dans un contexte donné, différenciés par leur position relative au mot à désambiguïser** » (Yarowsky, 2000 ; Yarowsky, Cucerzan, Florian, Schafer & Wicentowski, 2001 ; Ng & Lee, 1996 ; Ng, 1997 ; Ng & Lee, 2002 ; Pedersen, 2001 ; Escudero, Marquez & Rigau, 2000c, 2000b)
   Les *mots* peuvent prendre cinq formes :
   (a) le mot littéralement (comme il apparaît dans le texte) ;

(b) le lemme du mot ;

(c) l'étiquette morphosyntaxique du mot ;

(d) la classe d'objet du mot (par exemple *nom_de_pays*) ;

(e) une réponse à une question sur le mot (par exemple « le mot est-il écrit en lettres capitales ? »).

⋆ Yarowsky (2000) utilise les cinq formes.

⋆ Yarowsky *et al.* (2001) utilisent les formes 6a, 6b et 6c.

⋆ Ng et Lee (1996) utilisent uniquement la forme 6c.

⋆ Ng (1997) utilise uniquement la forme 6a, le contexte est de quatre mots en position -2, -1, $+1$, $+2$.

⋆ Ng et Lee (2002) utilisent les formes 6a et 6c.

⋆ Pedersen (2001) utilise uniquement la forme 6a, le contexte est -1 et $+1$ (mots adjacents au mot à désambiguïser) et les cooccurrences ne sont considérées que si elles apparaissent au moins deux fois.

⋆ Escudero *et al.* (2000c, 2000b) utilisent les formes 6a et 6c, le contexte est, pour la forme 6a, de quatre mots en position -2 [1], -1, $+1$, $+2$ [1] et, pour la forme 6c, de six mots en position -3, -2, -1, $+1$, $+2$, $+3$.

⋆ Habituellement, quand ce sont les étiquettes morphosyntaxiques du mot qui sont considérées, les mots qui se trouvent en dehors de la phrase sont rejetés.

7. « **mots situés dans un contexte donné sans tenir compte de leur position relative au mot à désambiguïser** » (Yarowsky, 2000 ; Yarowsky, Cucerzan, Florian, Schafer & Wicentowski, 2001 ; Ng & Lee, 1996, 2002 ; Pedersen, 2001 ; Mooney, 1996 ; El-Bèze, Loupy & Marteau, 1998 ; El-Bèze, Michelon & Pernaud, 1999)

Ce critère est parfois désigné en anglais par *bag of words* ou encore par *unordered set of surrounding words*.

⋆ Dans les travaux de Yarowsky (2000) et de Yarowsky *et al.* (2001) les *mots* peuvent prendre les différentes formes présentées pour le critère précédent (critère 6).

⋆ Dans les travaux de Ng et Lee (1996) les *mots* prennent la forme littérale des mots du texte (forme 6a). Ce sont des mots clefs présélectionnés par une méthode de filtrage qui ne retient qu'un nombre limité (cinq par lexies dans l'expérience réalisée) de cooccurrences jugées les plus pertinentes pour la désambiguïsation.

⋆ Dans les travaux de Ng et Lee (2002) les *mots* prennent la forme de la racine des mots du texte. Les mots appartenant à une liste de mots vides (prépositions, articles, etc. *stop words* en anglais) ou ne contenant pas de caractère alphabétique (nombres, ponctuations, etc.) ne sont pas considérés.

⋆ Pedersen (2001) utilise seulement la forme 6a et les cooccurrences ne sont considérées que si elles apparaissent au moins cinq fois.

⋆ Dans les travaux de Mooney (1996) les *mots* prennent la forme littérale des mots du texte (forme 6a). Le contexte est constitué par les mots de la phrase contenant le mot à désambiguïser et ceux de la phrase précédente. Tous les mots sont convertis en minuscules et les mots très fréquents (*stop words* en anglais) sont ignorés. Il faut remarquer que c'est le seul critère utilisé dans cette étude.

⋆ Dans les travaux de El-Bèze *et al.* (1998, 1999) les *mots* correspondent aux lemmes des mots du texte (forme 6b). Le contexte est constitué par les trois mots

1. Seulement utilisé dans (Escudero *et al.*, 2000c) et uniquement pour le premier ensemble de critères (cf. section 7.2.2).

qui suivent et qui précèdent le mot à désambiguïser, les adjectifs possessifs, les déterminants, les adverbes et les verbes ne sont pas considérés.

n-grammes avec $n > 1$

Dans le cadre de notre étude, nous définissons le terme *n-gramme* de la manière suivante :

Définition 7.2 – n-gramme –
Un n-gramme est une portion de texte constituée de n *mots consécutifs.*

Cette définition est à pondérer par le fait que certaines équipes n'incluent pas le mot à désambiguïser dans le n-gramme. Voici le critère basé sur les n-grammes utilisé par d'autres équipes dans le cadre d'études similaires à la nôtre :

8. « **n-grammes** » (Yarowsky, 2000 ; Yarowsky *et al.*, 2001 ; Ng & Lee, 1996 ; Ng, 1997 ; Ng & Lee, 2002 ; Pedersen, 2001 ; Escudero *et al.*, 2000c, 2000b)
   ⋆ Dans les travaux de Yarowsky (2000) et Yarowsky *et al.* (2001), les *mots* qui composent les *n-grammes* peuvent prendre les différentes formes présentées pour le critère 6.
   ⋆ Dans les travaux de Ng et Lee (1996), les *mots* qui composent les *n-grammes* prennent la forme littérale des mots du texte (forme 6a) et contiennent obligatoirement le mot à désambiguïser. Dans cette expérience, seuls les neuf *n-grammes* jugés les plus pertinents pour la désambiguïsation sont retenus.
   ⋆ Dans les travaux de Ng (1997), les *mots* qui composent les *n-grammes* prennent la forme littérale des mots du texte (forme 6a). Trois n-grammes sont considérés avec les combinaisons de mots suivantes (la position relative du mot par rapport au mot à désambiguïser est notée en indice) : $(mot_{-1} - mot_{+1})$, $(mot_{-2} - mot_{-1})$, $(mot_{+1} - mot_{+2})$.
   ⋆ Dans les travaux de Ng et Lee (2002), les *mots* qui composent les *n-grammes* prennent la forme littérale des mots du texte (forme 6a) mais ne contiennent pas obligatoirement le mot à désambiguïser.
   ⋆ Dans les travaux de Pedersen (2001), les *mots* qui composent les *n-grammes* prennent la forme littérale des mots du texte (forme 6a). Les *n-grammes* considérés sont en fait des bigrammes ($n = 2$) filtrés par une contrainte de fréquence et de valeur minimale de la mesure des logarithmes des probabilités (*log-likelihoods*).
   ⋆ Dans les travaux de Escudero *et al.* (2000c, 2000b), les *mots* qui composent les *n-grammes* prennent la forme littérale des mots du texte (forme 6a). Sept n-grammes sont considérés avec les combinaisons de mots suivantes (la position relative du mot par rapport au mot à désambiguïser est notée en indice) : $(mot_{-2} - mot_{-1})$, $(mot_{-1} - mot_{+1})$, $(mot_{+1} - mot_{+2})$, $(mot_{-3} - mot_{-2} - mot_{-1})$, $(mot_{-2} - mot_{-1} - mot_{+1})$, $(mot_{-1} - mot_{+1} - mot_{+2})$, $(mot_{+1} - mot_{+2} - mot_{+3})$.
   ⋆ Habituellement, dans un n-grammes, les mots qui se trouvent en dehors de la phrase sont rejetés.

Relations syntaxiques

Voici une liste non exhaustive de critères basés sur des relations syntaxiques binaires :

9. « **mot en relation sujet-verbe** » (Yarowsky, 1993, 2000 ; Yarowsky, Cucerzan, Florian, Schafer & Wicentowski, 2001)

Ce critère est applicable aux noms et aux verbes. Il retourne, quand c'est possible, le sujet du verbe quand le mot à désambiguïser est un verbe et le sujet un nom, et le verbe dont le nom est sujet quand le mot à désambiguïser est un nom sujet d'un verbe.

⋆ Dans les travaux de Yarowsky (2000), les « mots » retournés par ce critère peuvent prendre les différentes formes présentées pour le critère 6.

⋆ L'étude de Ng et Lee (1996) ne porte que sur le nom *interest*, ce critère n'est donc utilisé que pour un nom et seul les verbes jugés les plus pertinents pour la désambiguïsation du nom sont candidats.

10. « **mot en relation verbe-objet** » (Yarowsky, 1993, 2000 ; Yarowsky *et al.*, 2001 ; Ng & Lee, 1996)
    Ce critère est analogue au critère précédent (9) mais pour une relation *verbe-objet*.
11. « **mot en relation adjectif-nom** » (Yarowsky, 1993, 2000 ; Yarowsky *et al.*, 2001)
    Ce critère est analogue aux critères précédents (9 et 10) mais pour une relation *adjectif-nom*. Il n'est utilisé que pour les adjectifs dans (Yarowsky, 1993).
12. « **mot en relation nom-nom** » (Yarowsky, 2000 ; Yarowsky *et al.*, 2001)
    Ce critère est analogue aux critères précédents (9, 10 et 11) mais pour une relation *nom-nom*.

Ng et Lee (2002) explorent des critères complexes (critères 13, 14 et 15) basés sur des relations syntaxiques. Grâce à un analyseur statistique (Charniak, 2000) et à un traitement subséquent, chaque mot pointe sur un mot parent. Par exemple, dans la phrase *Reid saw me looking at the iron bars*, les mots *Reid* et *me* pointent sur le même parent *saw*.

13. Dans le cas où le mot w à désambiguïser est un nom, quatre informations sont retournées :
    - la forme littérale du parent ;
    - l'étiquette morphosyntaxique du parent ;
    - la forme passive ou active du parent quand il s'agit d'un verbe ;
    - la position relative du parent (à droite ou à gauche du mot w).
14. Dans le cas où le mot w à désambiguïser est un verbe, six informations sont retournées :
    - la forme littérale du mot le plus proche l situé à gauche du mot w tel que w soit le parent de l ;
    - la forme littérale du mot le plus proche r situé à droite du mot w tel que w soit le parent de r ;
    - l'étiquette morphosyntaxique de l ;
    - l'étiquette morphosyntaxique de r ;
    - l'étiquette morphosyntaxique de w ;
    - la voie (passive ou active) de w.
15. Dans le cas où le mot w à désambiguïser est un adjectif, deux informations sont retournées :
    - la forme littérale du parent ;
    - l'étiquette morphosyntaxique du parent.

7.2.2 Étude comparative des critères déjà réalisée par d'autres équipes

Les études comparatives de critères sont relativement rares. En fait la plupart des travaux dans le domaine utilisent un critère unique, ou combinent plusieurs critères, sans justifier le choix de ces critères. Il est certain que ce choix a dû faire l'objet d'une étude préliminaire plus ou moins empirique et plus ou moins exhaustive, mais elle est rarement publiée. Certaines études comparatives de critères ont toutefois été menées et publiées. Nous présentons les résultats de quelques-unes de ces études dans ce qui suit.

Yarowsky (1993) effectue une étude comparative des critères 2, 3, 4, 5, 9, 10 et 11 qui ne retournent au maximum qu'un mot du contexte. Il s'agit de montrer qu'un mot à désambiguïser ne possède qu'un sens pour une cooccurrence donnée. Cette étude donne de bons résultats (précision comprise entre 90% et 99%) validant ainsi l'hypothèse et l'intérêt des critères utilisés. Il faut cependant modérer les résultats car les mots utilisés sont des *pseudo-mots* (cf. section 2.5.3) ne comportant que deux lexies. Yarowsky tire également les conclusions suivantes des résultats obtenus :

- les verbes sont plus efficacement désambiguïsés par leur objet que par leur sujet ;
- les adjectifs sont très majoritairement désambiguïsés par le nom qu'ils modifient ;
- les noms sont majoritairement désambiguïsés par le nom ou l'adjectif qui leur est directement adjacent ;
- les verbes ne sont pas très utiles pour lever l'ambiguïté des noms et ils le sont plus quand le nom est leur objet plutôt que leur sujet.

Dans cette étude, Yarowsky observe que l'information pour la désambiguïsation est bien plus locale pour les verbes et les adjectifs que pour les noms. Il observe également que les mots grammaticaux apportent de l'information quand ils sont directement adjacents au mot à désambiguïser.

Ng et Lee (1996) effectuent une petite étude comparative des critères 6, 7, 8 et 9 et observent que le critère 8 est le plus performant (précision de 80,2%) suivi, dans l'ordre de performance décroissante, des critères 6 (77,2%), 7 (62%) et 9 (43,5%). L'utilisation conjointe de ces quatre critères permet d'atteindre une précision de 87,4%. Il faut remarquer que cette étude porte sur le seul mot *interest* et sur une version restrictive de ces critères.

Ng et Lee (2002) étudient plusieurs critères :

- un critère nommé *POS* correspondant au critère 6 en utilisant la forme 6c;
- un critère nommé *surrounding words* correspondant au critère 7 ;
- un critère nommé *Collocations* regroupant le critère 6 en utilisant la forme 6a et le critère 8 ;
- un critère nommé *Syntactic relations* regroupant les critères 13, 14 et 15.

Ces critères sont évalués tels quels et en filtrant les indices en fonction de leur fréquence par lexies. D'autre part, Ng et Lee utilisent quatre méthodes de classification et étudient les interactions qui peuvent survenir entre le choix des critères et celui des méthodes de classification. Leurs conclusions sont les suivantes :

- le critère, évalué individuellement, obtenant les meilleures performances est fonction de la méthode de classification utilisée ;
- le filtrage des indices dégrade les performances pour trois des quatre méthodes de classification évaluées ;
- l'utilisation conjointe des quatre critères augmente les performances pour trois des quatre méthodes de classification évaluées.

Cette étude suggère qu'il existe une forte interaction entre critères et méthodes de classification.

Dans le cadre de la campagne d'évaluation SensEval-2, Pedersen (2001) évalue un grand nombre de combinaisons de critères et de méthodes de classification. Son expérience lui permet de conclure que les variations des performances dues aux choix des méthodes de classification sont bien plus faibles que celles dues au choix des critères. En d'autres termes, un choix pertinent de critères donne de bonnes performances avec une grande variété de méthodes de classification, tandis qu'aucune méthode de classification ne peut pallier les limitations dues à un mauvais choix de critère.

Escudero, Marquez et Rigau (2000c) évaluent deux ensembles de critères, le premier correspond au critère 6 restreint à la forme 6a, le second correspond aux critères 6 et 8. L'évaluation est menée avec deux méthodes de classification, la première est du type classifieur naïf de Bayes et la seconde du type k plus proches voisins. Dans cette étude, le deuxième ensemble de critère, bien plus riche que le premier, obtient de meilleures performances avec la méthode du type k plus proches voisins, mais de moins bonnes avec le classifieur naïf de Bayes.

7.2.3 Performances atteintes par d'autres équipes

Il est intéressant de savoir quels sont les critères utilisés par d'autres équipes. Néanmoins, il est souvent difficile de comparer entre elles les performances obtenues par d'autres équipes, même en se restreignant aux méthodes supervisées. Cette difficulté de comparaison se transforme même en quasi-impossibilité avec notre étude et cela pour plusieurs raisons :

- certaines études traitent le problème de l'ambiguïté catégorielle, d'autres non ; notre étude considère que ce problème fait l'objet d'un prétraitement indépendant de celui de la désambiguïsation lexicale ;
- toutes les études n'utilisent pas les mêmes corpus ; dans notre étude, nous utilisons un corpus constitué dans le cadre du projet SyntSem, en cours de constitution, utilisé pour la première fois dans le cadre d'une étude en désambiguïsation lexicale ;
- toutes les études n'utilisent ni les mêmes dictionnaires ni la même granularité au niveau des sens ; dans notre étude, nous utilisons un dictionnaire distributionnel, en cours de constitution, utilisé pour la première fois dans le cadre d'une étude en désambiguïsation lexicale avec une granularité fine au niveau des sens ;
- toutes les études ne portent pas sur les mêmes mots, certaines ne portent que sur un ou peu de mots ; certaines se limitent à une seule classe grammaticale alors que d'autres ont une couverture plus large ; notre étude porte sur 60 mots répartis en 3 classes grammaticales (nom, adjectif et verbe) ;
- toutes les études ne portent pas sur la même langue ; alors que la majorité des études est réalisée sur l'anglais, la nôtre porte sur le français.

De manière à pouvoir comparer rigoureusement les performances des algorithmes de désambiguïsation lexicale, une campagne d'évaluation a été créée : la campagne d'évaluation SensEval. Une première campagne d'évaluation, SensEval-1, est réalisée en 1998. Au niveau de la désambiguïsation lexicale supervisée seule la langue anglaise est alors concernée. Une deuxième campagne d'évaluation, SensEval-2, est réalisée en 2001. Au niveau de la désambiguïsation lexicale supervisée, les langues concernées sont l'anglais, le basque, l'italien, le coréen, l'espagnol et le suédois. À titre indicatif, le tableau 7.1 donne les meilleures performances obtenues par un algorithme supervisé,

		SensEval-1	SensEval-2
Moyenne	Précision du meilleur	76,0%	64,2%
	Rappel du meilleur	75,1%	64,2%
	Précision de majoritaire	59,1%	
	Entropie des lexies	1,837	
Noms	Précision du meilleur	83,3%	69,5%
	Rappel du meilleur	83,3%	
	Précision de majoritaire	57,3%	
	Entropie des lexies	1,892	
Adjectifs	Précision du meilleur	76,6%	73,2%
	Rappel du meilleur	73,8%	
	Précision de majoritaire	64,3%	
	Entropie des lexies	1,7	
Verbes	Précision du meilleur	70,5%	57,6%
	Rappel du meilleur	69,7%	
	Précision de majoritaire	57,9%	
	Entropie des lexies	1,859	

Tableau 7.1 – Meilleures performances obtenues par un algorithme supervisé dans le cadre des campagnes d'évaluation SensEval-1 et SensEval-2.

pour l'anglais, dans le cadre des campagnes d'évaluation SensEval-1 et SensEval-2. Il est intéressant de noter que si les résultats obtenus par diverses équipes au sein d'une même campagne peuvent être comparés, il est difficile de comparer les résultats entre les campagnes SensEval-1 et SensEval-2.

7.2.4 Protocole expérimental

Choix des classifieurs

Pour l'étude des critères de désambiguïsation lexicale automatique, nous utilisons principalement deux classifieurs : NB(0,00) et PCM(0,00). Les raisons de ce choix sont multiples :

- ce sont deux classifieurs performants ;
- ils classifient le maximum d'occurrences possibles ;
- ils sont très utilisés en désambiguïsation lexicale et font partie des références en la matière ;
- ils sont rapides, caractéristique primordiale étant donnée la quantité importante d'expériences que nous mènerons ;
- ils sont complémentaires puisque l'un combine les attributs et l'autre non ;
- le classifieur PCM(0,00) utilise une liste de décisions ce qui rend la procédure de décision transparente et permet des observations intéressantes sur les indices sélectionnés pour la prise de décision.

Ces deux classifieurs effectuent leur classification sur pratiquement toutes les occurrences (plus précisément sur 99,93% des occurrences). Pour pouvoir les comparer plus facilement, sans avoir recours à la mesure de performance, nous nous ramenons à deux classifieurs qui effectuent une classification sur toutes les instances à classer en affectant à toutes les instances non classées la classe majoritaire. Pour les nommer, nous rajoutons un T (pour Total) devant le nom du classifieur dont ils sont issus : TNB(0,00) et TPCM(0,00).

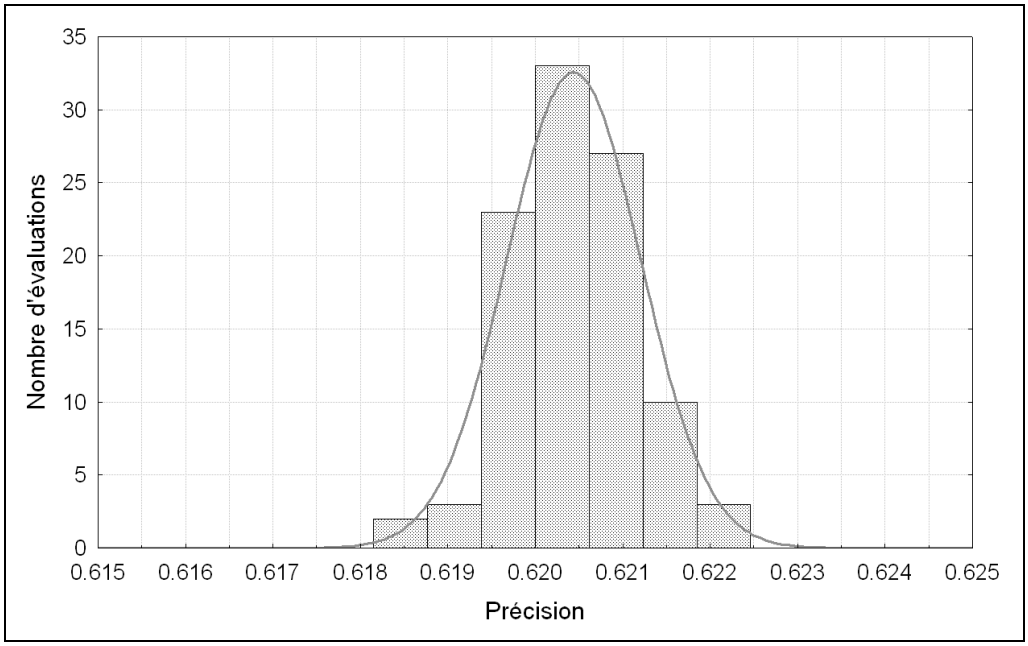


Figure 7.1 – Dispersion, pour cent évaluations, des précisions moyennes du classifieur TPCM(0,00) pour les 60 vocables.

Validation croisée k-fois

Comme nous l'avons fait pour l'évaluation de quelques classifieurs dans le chapitre 6, nous utilisons la méthode de validation croisée k-fois (cf. section 6.2.5) pour estimer les performances de nos algorithmes. Pour cela nous devons partitionner aléatoirement notre corpus (dans ce chapitre, nous utilisons l'intégralité du corpus) en k sous-ensembles. Une question se pose alors, effectuons-nous ce partitionnement aléatoire pour chaque évaluation ou une fois pour toutes ?

Nous avons réalisé une centaine d'évaluations en effectuant le partitionnement aléatoirement pour chacune. La figure 7.1 montre la dispersion des précisions moyennes obtenues pour le classifieur TPCM(0,00) et la figure 7.2 montre la dispersion des précisions moyennes obtenues pour le classifieur TNB(0,00). Ces deux dispersions semblent suivre une loi normale.

Le tableau 7.2 synthétise les résultats obtenus. Pour les deux algorithmes, l'écart-type des précisions obtenues est inférieur à 0,08% et la différence entre la précision maximale et la précision minimale obtenue est inférieure à 0,5%. Les fluctuations des résultats obtenus en fonction des partitionnements de l'ensemble d'apprentissage sont très faibles. Les résultats obtenus pour un partitionnement donné sont donc représentatifs. Pour la suite de l'étude, nous choisissons un partitionnement aléatoire une fois pour toutes pour deux raisons :

- cette façon de procéder facilite la comparaison des performances de différents critères en s'affranchissant des fluctuations engendrées par des partitionnements différents, la cohérence des résultats de toute l'étude s'en trouve augmentée ;
- cette façon de procéder permet également de reproduire facilement une expérience pour vérifier la validité des résultats obtenus.

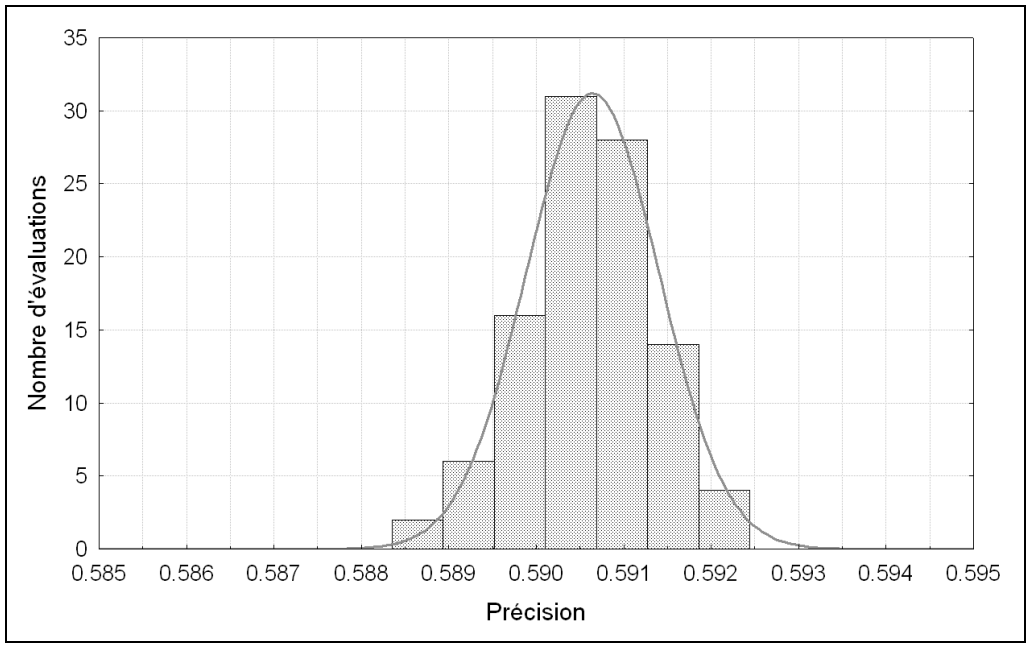


Figure 7.2 – Dispersion, pour cent évaluations, des précisions moyennes du classifieur TNB(0,00) pour les 60 vocables.

	Moyenne	Maximum	Minimum	Max - Min	Écart-type
TPCM(0,00)	62,0%	62,2%	61,8%	0,43%	0,076%
TNB(0,00)	59,1%	59,2%	58,8%	0,41%	0,075%

Tableau 7.2 – Synthèse des résultats des différentes précisions obtenues en fonction des partitionnements.

Automatisation des traitements

Pour automatiser une série de traitements, nous utilisons des scripts qui s'appellent mutuellement. Cette automatisation des séries de traitement impose que nous écrivions des règles paramétrables pour modéliser les critères étudiés. En fait, chaque règle possède au moins deux paramètres : un paramètre concernant la taille du contexte étudié et un paramètre précisant le vocable sur lequel porte l'étude.

7.3 Critères basés sur les cooccurrences évalués indépendamment

7.3.1 Introduction

De nombreuses études (Kelly & Stone, 1975 ; Yarowsky, 1993 ; Mooney, 1996 ; Ng & Lee, 1996 ; Agirre & Martinez, 2001a ; etc.) montrent que les cooccurrences constituent un bon critère pour identifier le sens d'un mot. Dans cette étude, nous nous proposons d'étudier de manière systématique des critères élémentaires, basés sur les cooccurrences, sans chercher à les combiner. Notre objectif est de fournir des informations de référence pour l'élaboration de critères plus complexes. Un critère basé sur les cooccurrences peut prendre de multiples formes en fonction des réponses apportées aux questions suivantes :

- Quelle est la taille optimale du contexte dans lequel les cooccurrences sont prises en compte ?
- Faut-il accepter ou pas de sortir de la phrase ?
- Faut-il filtrer ou pas les mots considérés (par exemple, ne retenir que les mots pleins ou au contraire prendre en compte tous les mots, même les mots grammaticaux) ?
- La cible doit-elle être traitée comme les autres mots du contexte ?
- Faut-il considérer indifféremment tous les mots (*unordered set of surrounding words* ou *bag of words* en anglais) ?
  Faut-il distinguer les mots apparaissant dans le contexte droit des mots apparaissant dans le contexte gauche ?
  Ou bien faut-il distinguer les mots en fonction de leur position par rapport au mot cible ?
- Il faut enfin choisir l'une des étiquettes des mots : la forme fléchie, le lemme, l'étiquette morphosyntaxique ou encore l'étiquette morphosyntaxique simplifiée.

Remarque : Par la suite, lorsque nous parlons d'une taille de contexte, il s'agit toujours d'une taille de contexte en nombre de mots considérés. Ainsi, quand nous parlons d'une taille de fenêtre de plus ou moins deux mots, si nous ne considérons que les mots pleins, nous entendons par là les deux mots pleins à droite et les deux mots pleins à gauche, et non les mots pleins (probablement de l'ordre de un ou deux) qui se trouvent dans les quatre mots qui entourent le mot à désambiguïser.

7.3.2 Faut-il accepter de sortir de la phrase ?

Cette question se pose quand un critère porte sur un petit contexte. Dans un tel cas, un grand nombre d'études rejettent les mots qui se trouvent en dehors de la phrase. Pour vérifier la pertinence d'une telle démarche, nous évaluons les performances de trois critères et comparons les performances obtenues avec celles de trois critères équivalents

mais qui rejettent les mots se trouvant en dehors de la phrase. Nous pouvons énoncer ces critères de la manière suivante :

- « **lemmes, différenciés par leur position par rapport au mot à désambiguïser, des n mots pleins (nom, adjectif, adverbe ou verbe) qui suivent ou qui précèdent le mot à désambiguïser** » ; nous notons ce critère *[lemme]-[ordonne]-[mot-plein]** ; la variante où les mots qui se trouvent en dehors de la phrase sont rejetés est notée *[lemme]-[ordonne]-[mot-plein]*.
- « **formes brutes, différenciées par leur position par rapport au mot à désambiguïser, des n mots qui suivent ou qui précèdent le mot à désambiguïser** » ; nous notons ce critère *[jeton]-[ordonne]-[mot]** ; la variante où les mots qui se trouvent en dehors de la phrase sont rejetés est notée *[jeton]-[ordonne]-[mot]* ;
- « **étiquettes morphosyntaxiques, différenciées par leur position par rapport au mot à désambiguïser, des n mots qui suivent ou qui précèdent le mot à désambiguïser** » ; nous notons ce critère *[ems]-[ordonne]-[mot]** ; la variante où les mots qui se trouvent en dehors de la phrase sont rejetés est notée *[ems]-[ordonne]-[mot]* ;

Nous mesurons les performances obtenues avec ces critères en faisant varier n entre un et six. Prenons par exemple l'extrait de corpus suivant :

> *La Commission a l ' intention de résoudre cette question de concert avec celle de l ' harmonisation des procédures d ' inspection et de* **détention** *des navires . Selon les informations dont la Commission dispose , il semblerait que certains assureurs . . .*

Pour $n = 5$, et en supposant que le mot à désambiguïser est *détention*, les critères retournent les descriptions suivantes :

***[lemme]-[ordonne]-[mot-plein]** retourne la description :** {question, concert, harmonisation, procédure, inspection, navire, information, commission, disposer, sembler}.

***[lemme]-[ordonne]-[mot-plein]* retourne la description :** {question, concert, harmonisation, procédure, inspection, navire}.

***[jeton]-[ordonne]-[mot]** retourne la description :** {(-5)d, (-4)', (-3)inspection, (-2)et, (-1)de, (+1)des, (+2)navires, (+3)., (+4)Selon, (+5)les}.

***[jeton]-[ordonne]-[mot]* retourne la description :** {(-5)d, (-4)', (-3)inspection, (-2)et, (-1)de, (+1)des, (+2)navires, (+3).}.

***[ems]-[ordonne]-[mot]** retourne la description :** {(-5)PREP, (-4)PREP, (-3)NCFS, (-2)COO, (-1)PREP, (+1)DETDPIG, (+2)NCMP, (+3)PCTFORTE, (+4)PREP, (+5)DETDPIG}.

***[ems]-[ordonne]-[mot]* retourne la description :** {(-5)PREP, (-4)PREP, (-3)NCFS,(-2)COO,(-1)PREP,(+1)DETDPIG,(+2)NCMP,(+3)PCTFORTE}.

Nous pouvons ainsi comparer les performances obtenues par les trois critères qui considèrent les mots même en dehors de la phrase avec leurs trois équivalents qui rejettent les mots en dehors de la phrase. En comparant les résultats moyens obtenus en fonction des trois catégories de vocables (noms, adjectifs et verbes), des trois types de critères et des six valeurs de n (de $n = 1$ à $n = 6$), nous obtenons $3 \times 3 \times 6$ soit 54 points de comparaison. En utilisant l'algorithme TPCM(0,00), nous observons qu'accepter les mots en dehors de la phrase n'apporte rien ou dégrade les performances dans les 54 cas. En utilisant l'algorithme TNB(0,00), accepter les mots en dehors de la phrase permet d'améliorer les performances dans seulement 3 cas sur 54. Dans tous les cas, les per-

formances sont très proches. La dégradation de la précision en acceptant les mots en dehors de la phrase est souvent de l'ordre de 0,3% et rarement supérieure à 1%.

Pour savoir si le signe de ponctuation forte de fin de phrase doit être considéré comme un mot faisant partie de la phrase, nous avons comparé les performances des trois critères qui rejettent les mots se trouvant en dehors de la phrase avec celles de trois critères équivalents qui rejettent également le signe de ponctuation forte de fin de phrase. Les résultats obtenus montrent que les trois premiers critères (qui prennent en compte le signe de ponctuation forte de fin de phrase) donnent de meilleurs résultats (la précision est supérieure de 0.2% à 0.5%). Pour illustrer ce phénomène, prenons le vocable *détention* avec ses deux lexies, $L1$ pour le sens incarcération ou enfermement de détention, et $L2$ pour le sens possession de détention (cf. annexe C.4). Bien que ce ne soit pas une obligation, les occurrences de la lexie $L2$ attendent généralement un argument après le vocable *détention* :

- « . . . la **détention** d ' armes . . . » ;
- « . . . le régime d ' acquisition et de **détention** de munitions . . . » ;
- « . . . procédures d ' inspection et de **détention** des navires . . . » ;
- « . . . une autorisation de **détention** d ' un fusil de chasse . . . » .

L'occurrence *détention* suivi d'une ponctuation forte incite à penser qu'il s'agit de *détention* au sens $L1$:

- « . . . racontent leur longue **détention** . » ;
- « . . . les deux quartiers de **détention** . » ;
- « . . . son inculpation et sa mise en **détention** . ».

Dans notre corpus, se trouvent 6 occurrences de la forme *détention* suivies d'une ponctuation forte et pour ces 6 occurrences la lexie de *détention* est $L1$.

En conclusion, il semble que les meilleures performances soient atteintes en se limitant aux mots de la phrase et en incluant le signe de ponctuation forte de fin de phrase dans ces mots. Cependant, en ne respectant pas ces contraintes, la dégradation des performances est modérée.

7.3.3 Faut-il incorporer le mot à désambiguïser ?

Le mot à désambiguïser peut-il apporter de l'information quant à sa désambiguïsation ? Si nous nous intéressons à son lemme (il est unique pour un vocable donné) il est peu probable que l'information apportée soit importante. La prise en compte du lemme du mot à désambiguïser apporte tout de même une information sur la fréquence relative des lexies. En effet, cette information génère un indice dont la répartition suivant les lexies correspond justement à la répartition de la fréquence de ces lexies. Cette information incite aux choix de la lexie majoritaire, ce qui reste le meilleur choix possible en l'absence d'indice plus pertinent. Contrairement au lemme, la forme brute ou l'étiquette morphosyntaxique peut prendre plusieurs formes pour un vocable donné et apporter ainsi une information susceptible d'être plus favorable pour la levée de son ambiguïté. Pour savoir ce que peut apporter le mot à désambiguïser, nous reprenons les trois critères de la section précédente (*[lemme]-[ordonne]-[mot-plein]*, *[jeton]-[ordonne]-[mot]* et *[ems]-[ordonne]-[mot]*) et nous les comparons avec leurs homologues incorporant l'information du mot à désambiguïser.

Le tableau 7.3 indique, pour chacun des trois critères et pour chacune des trois catégories de vocables, si la prise en compte du mot à désambiguïser s'est traduite par une augmentation de la précision (+) ou par une dégradation (-) de cette dernière. La partie supérieure du tableau reflète les augmentations et dégradations pour le classifieur

		Nom	Adjectifs	Verbes	**Moyenne**	Gain
TPCM(0,00)	[lemme]-[ordonne]-[mot-plein]	-	-	-	**-**	−0,6%
	[jeton]-[ordonne]-[mot]	-	-	-	**-**	−0,2%
	[ems]-[ordonne]-[mot]	-	-	+	+	0,2%
TNB(0,00)	[lemme]-[ordonne]-[mot-plein]	-	+	+	+	0,2%
	[jeton]-[ordonne]-[mot]	+	+	+	+	0,8%
	[ems]-[ordonne]-[mot]	+	+	+	+	1,4%

Tableau 7.3 – Gain (+) ou dégradation (−) dus à la prise en compte du mot à désambiguïser.

TPCM(0,00) et la partie inférieure pour le classifieur TNB(0,00). La dernière colonne précise le gain moyen en précision obtenu en prenant en compte le mot à désambiguïser.

Les deux classifieurs testés réagissent de manière très différente : alors que pour le classifieur TPCM(0,00) la prise en compte du mot à désambiguïser se traduit plutôt par une dégradation, elle se traduit par une amélioration substantielle pour le classifieur TNB(0,00). Il faut également remarquer que l'amélioration la plus importante de la précision pour les deux classifieurs est obtenue avec le critère *[ems]-[ordonne]-[mot]*.

7.3.4 Définitions, formalisation et évaluation de 24 critères

Définitions des 24 critères

Comme nous l'avons vu dans la section 7.3.1, un critère basé sur les cooccurrences peut prendre de multiples formes. Nous allons comparer 24 familles de critères que nous désignons de la manière suivante : *[<param1>]-[<param2>]-[<param3>]*. *<param3>* peut prendre deux valeurs, *mot* et *mot-plein*, suivant que le critère considère tous les mots ou seulement les mots pleins. *<param2>* peut prendre trois valeurs, *ordonne*, *differencie* et *non-ordonne* selon que le critère tienne compte de l'ordre des mots (*ordonne*), que le critère différencie simplement le contexte droit du gauche (*differencie*) ou que le critère ne tienne pas compte de la position des mots (*non-ordonne*). Enfin, *<param1>* peut prendre quatre valeurs, *jeton*, *lemme*, *ems* et *smallems* suivant que le critère s'intéresse à la forme brute des mots (*jeton*), à leur lemme (*lemme*), à leur étiquette morphosyntaxique (*ems*) ou à leur étiquette morphosyntaxique simplifiée (*smallems*).

Au total, cela fait $4 \times 3 \times 2 = 24$ combinaisons différentes (*<param1>* peut prendre quatre valeurs, *<param2>* trois et *<param3>* deux), soit 24 critères.

L'étiquette *jeton* est celle qui présente la plus grande variabilité. Pour certains mots, il peut y avoir une ambiguïté sur la catégorie grammaticale de *jeton*, par exemple le *jeton porte* peut correspondre à un nom commun, un verbe ou un adjectif. Il est clair que cette ambiguïté catégorielle ne facilite pas la désambiguïsation. Pour supprimer cette ambiguïté sur les étiquettes *jeton*, nous accolons à l'étiquette *jeton* l'étiquette *ems*. Ainsi, quand nous parlons de l'étiquette *jeton*, il s'agit en réalité de la juxtaposition de l'étiquette *ems* et de l'étiquette *jeton*. D'après nos expériences, pour un contexte de plus ou moins deux mots et avec le critère *[jeton]-[ordonne]-[mot]* le gain obtenu par cette juxtaposition, par rapport à l'utilisation de l'étiquette *jeton* seule, est de l'ordre de 0,5% pour le classifieur TPCM(0,00) et de l'ordre de 0,3% pour le classifieur TNB(0,00).

Nous avons vu dans la section 7.3.2 qu'ignorer les mots qui ne se trouvent pas dans la phrase du mot à désambiguïser permettait une augmentation des performances, faible mais quasi-systématique. Aussi ignorerons-nous les mots hors de la phrase pour chacun des 24 critères. Dans la section 7.3.3, nous avons vu qu'incorporer le mots à désambi-

guïser dans le critères *[ems]-[ordonne]-[mot]* peut améliorer la désambiguïsation, mais que cette tendance ne se vérifie par forcément quand le critère est basé sur une autre étiquette que l'étiquette morphosyntaxique. Aussi, pour les six critères du type *[ems]-[<param2>]-[<param3>]* nous incluons l'information de la cible.

Exemple de formalisation de critère

Nous allons évaluer ces critères pour des contextes allant de plus ou moins un mot à plus ou moins huit mots et pour les 60 vocables. Cela revient à évaluer $24 \times 8 \times 60 = 11\,520$ critères. Bien entendu, nous n'allons pas écrire 11 520 règles pour modéliser chacun de ces 11 520 critères. Nous allons écrire des règles génériques permettant de modéliser toute une famille de critères. Ces règles doivent donc être paramétrables, nous utilisons trois paramètres :

1. Le premier paramètre, `@(1)`, permet de préciser le vocable sur lequel nous travaillons. Nous écrivons ainsi une unique règle générique pour l'ensemble des 60 vocables, de cette manière, le nombre de règles à écrire est divisé par 60.
2. Le second paramètre, `@(2)`, précise la taille du contexte. Une taille de deux signifie que le critère utilise un contexte de plus ou moins deux mots (*i.e.* deux mots à gauche et deux mots à droite). Ce paramètre permet de diviser le nombre de règles à écrire par huit.
3. Le troisième paramètre, `@(3)`, correspond en fait à la caractéristique *<param1>* des critères (*i.e. jeton*, *lemme*, *ems* ou *smallems*). Le nombre de règles à écrire est ainsi divisé par quatre.

Pour simplifier l'écriture des règles, nous les subdivisons en trois sous-règles :

- une règle pour décrire les indices situés à gauche du mot à désambiguïser ;
- une règle pour décrire l'indice correspondant au mot à désambiguïser ;
- une règle pour décrire les indices situés à droite du mot à désambiguïser.

La seconde règle n'est utilisée que pour les critères qui considèrent le mot à désambiguïser (*i.e.* les critères de la forme *[ems]-[<param2>]-[<param3>]*).

Au final, pour modéliser les 11 520 critères correspondant aux 24 familles de critères, il nous faut écrire $\frac{11\,520}{60 \times 8 \times 4} \times 3 = 18$ règles.

Pour illustrer notre propos, nous donnons et commentons un exemple de la formalisation d'une famille de critères. Les figures 7.3, 7.4 et 7.5 précisent comment a été formalisé la famille de critères *[<param1>]-[ordonne]-[mot]*.

Commentons la règle de la figure 7.3 en supposant que nous travaillons sur le vocable détention (`@(1)` = *détention*), avec un contexte de plus ou moins huit mots (`@(2)` = 8) et avec la famille de critères *[lemme]-[ordonne]-[mot]* (`@(3)` = *lemme*). Nous allons traiter cet exemple sur la portion de corpus suivante :

> « Le juge , Jean - Marie Lion , a sanctionné cette erreur de trajectoire d ' une inculpation d ' homicide volontaire . Le boulanger a été inculpé de **détention** illégale d ' armes de quatrième catégorie . Cinq amis d ' Ali Rafa ont été inculpés de vols , dégradation volontaire et voies de fait pour . . . »

La MERE `[]{1,@(2)}` décrit n'importe quelle portion de corpus contenant un à huit mots consécutifs, puisque `@(2)` = 8 dans notre exemple. La MERE

```
cible:[lemme="@(1)" & lexie~"^[0-9]"]
```

décrit un mot dont le lemme est *détention*, puisque `@(1)` = *détention*, et dont la lexie commence par un chiffre (les étiquettes lexicales valides commencent par un chiffre, les

```
Requête :
  []{1,@(2)} <cible:[lemme="@(1)" & lexie~"^[0-9]"]>
  stop(ems="PCTFORTE")

Apparence :
  [P:cible.fichier]-[P:cible.paragraphe]-[P:cible.position]

Référence :
  [?:"@(3)"="jeton";J;[?:"@(3)"="lemme";L;
  [?:"@(3)"="ems";E;[?:"@(3)"="smallems";S;err]]]]
  OM_@(2)_([N:begin.index-cible.index])_
  [?:"@(3)"="jeton";[P:begin.ems]-;][P:begin.@(3)]

Discriminant : [P:cible.lexie]
```

Figure 7.3 – Exemple de règle pour décrire les indices situés à gauche du mot à désambiguïser pour les critères du type *[<param1>]-[ordonne]-[mot]*.

```
Requête :
  cible:[lemme="@(1)" & lexie~"^[0-9]"]

Apparence :
  [P:cible.fichier]-[P:cible.paragraphe]-[P:cible.position]

Référence :
  [?:"@(3)"="jeton";J;[?:"@(3)"="lemme";L;
  [?:"@(3)"="ems";E;[?:"@(3)"="smallems";S;err]]]]OM_@(2)_
  (0)_[?:"@(3)"="jeton";[P:cible.ems]-;][P:cible.@(3)]

Discriminant : [P:cible.lexie]
```

Figure 7.4 – Exemple de règle pour décrire l'indice concernant le mot à désambiguïser pour les critères du type *[ems]-[ordonne]-[mot]*.

```
Requête :
  cible:[lemme="@(1)" & lexie~"^[0-9]"] []{1,@(2)}
  stop(prev.ems="PCTFORTE" & begin.index != end.index)

Apparence :
  [P:cible.fichier]-[P:cible.paragraphe]-[P:cible.position]

Référence :
  [?:"@(3)"="jeton";J;[?:"@(3)"="lemme";L;
  [?:"@(3)"="ems";E;[?:"@(3)"="smallems";S;err]]]]
  OM_@(2)_ ([N:end.index-cible.index])_
  [?:"@(3)"="jeton";[P:end.ems]-;][P:end.@(3)]

Discriminant : [P:cible.lexie]
```

Figure 7.5 – Exemple de règle pour décrire les indices situés à droite du mot à désambiguïser pour les critères du type *[<param1>]-[ordonne]-[mot]*.

autres correspondent à des erreurs de lemmatisation comme nous l'avons précisé dans la section 4.4.1). Le mot décrit est appelé `cible`. Ainsi, la MER

```
[]{1,@(2)} <cible:[lemme="@(1)" & lexie~"^[0-9]"]>
```

décrit les portions de deux à neuf mots consécutifs dont le dernier mot est le vocable *détention*. Les `<>` qui entourent la seconde MERE précisent qu'il faut d'abord identifier le vocable *détention* avant de tenter de générer les huit correspondances possibles en ajoutant des mots à gauche, c'est-à-dire en continuant la recherche de correspondances en avançant à rebours dans le corpus à partir du mot *détention*. Notre MER retourne donc huit correspondances qui désignent les portions de corpus suivantes :

1. *de* ***détention*** ;
2. *inculpé de* ***détention*** ;
3. *été inculpé de* ***détention*** ;
4. *a été inculpé de* ***détention*** ;
5. *boulanger a été inculpé de* ***détention*** ;
6. *Le boulanger a été inculpé de* ***détention*** ;
7. *. Le boulanger a été inculpé de* ***détention*** ;
8. *volontaire . Le boulanger a été inculpé de* ***détention*** ;

La requête, de la règle de la figure 7.3, contient également la clause *stop* suivante :

```
stop(ems="PCTFORTE")
```

L'expression logique `ems="PCTFORTE"` est vraie quand l'étiquette morphosyntaxique du mot courant est *PCTFORTE*. La clause *stop* précise donc qu'il faut arrêter la recherche des correspondances dès que le mot courant est une ponctuation forte. Toujours sur la même portion de corpus, notre requête ne retourne plus que les six premières correspondances de la MER. En effet, lors de la recherche de la septième, le mot courant est une ponctuation forte, la clause *stop* est alors vraie et la tentative de recherche d'une nouvelle correspondance échoue.

La clause *stop* de la requête de la règle de la figure 7.5, est légèrement différente :

```
stop(prev.ems="PCTFORTE" & begin.index != end.index)
```

La raison de cette différence provient du fait que cette requête décrit les indices situés à droite du mot à désambiguïser. Dans ce cas, la ponctuation forte de fin de phrase fait partie de la phrase et doit être conservée. L'expression logique `prev.ems="PCTFORTE"` est vraie quand l'étiquette morphosyntaxique du mot précédent le mot courant est *PCTFORTE*. La seconde expression logique, `begin.index!=end.index`, est vraie quand la taille courante de la correspondance est strictement supérieure à un mot. Ainsi, la clause *stop* précise qu'il faut arrêter la recherche des correspondances quand le mot courant fait partie de la phrase suivante, ce qui est le cas quand :

- le mot précédent est une ponctuation forte (`next.ems="PCTFORTE"`),
- et la correspondance n'est pas réduite au seul mot à désambiguïser (`begin.index!=end.index`).

Reprenons la description de la règle 7.3 là où nous l'avions interrompue. Dans l'extrait de corpus, le fichier est *M* (articles du Monde), le numéro du paragraphe est *10294* et la position du mot *détention* est *153*. Le masque *apparence*

```
Apparence :
[P:cible.fichier]-[P:cible.paragraphe]-[P:cible.position]
```

génère donc la chaîne `M-10294-153` pour chacune des sept correspondances.

Le masque *référence* est plus complexe :

```
Référence :
```

```
[?:"@(3)"="jeton";J;[?:"@(3)"="lemme";L;
[?:"@(3)"="ems";E;[?:"@(3)"="smallems";S;err]]]]
OM_@(2)_([N:begin.index-cible.index])_
[?:"@(3)"="jeton";[P:begin.ems]-;][P:begin.@(3)]
```

La première partie du masque (les deux premières lignes) génère la lettre *J* si `@(3)` = *jeton*, la lettre *L* si `@(3)` = *lemme*, la lettre *E* si `@(3)` = *lemme* et la lettre *S* si `@(3)` = *smallems*. Dans notre cas, cette première partie génère donc la lettre *L* puisque `@(3)` = *lemme*.
La partie suivante, `OM_@(2)_(`, génère tout simplement la chaîne *OM_ 8_ (*.
La partie `[N:begin.index-cible.index]` génère un nombre entier correspondant à la différence entre la position dans le corpus du premier mot de la correspondance et la position dans le corpus de la cible. Comme la MER décrit des portions de corpus à gauche de la cible, cette différence génère toujours un entier négatif pour cette règle.
La chaîne `)_` est ensuite générée.
`[?:"@(3)"="jeton";[P:begin.ems]-;]` génère l'étiquette morphosyntaxique du premier mot de la correspondance quand `@(3)` = *jeton* et rien sinon (ce qui est notre cas).
Enfin, la dernière partie, `[P:begin.@(3)]` génère l'étiquette `@(3)`, le lemme dans notre cas, du premier mot de la correspondance.
Les chaînes générées pour nos six correspondances sont donc respectivement :

1. `LOM_8_(-1)_de`
2. `LOM_8_(-2)_inculper`
3. `LOM_8_(-3)_être`
4. `LOM_8_(-4)_avoir`
5. `LOM_8_(-5)_boulanger`
6. `LOM_8_(-6)_le`

Il s'agit en fait des six indices générés à gauche pour notre critère. `LOM` est la signature du critère *[lemme]-[ordonne]-[mot]*. `_8` permet de préciser que le contexte est de plus ou moins huit mots. Le nombre entre parenthèses qui précède le lemme permet de différencier les mots en fonction de leur position. Si le critère était *[lemme]-[non-ordonne]-[mot]*, cette partie serait absente et le premier indice serait `LNoM_8_de`. Si le critère était *[lemme]-[differencie]-[mot]*, nous aurions la lettre `G` pour gauche et le premier indice serait `LDM_8_G_de`. La signature du critère nous permet de pouvoir combiner des critères sans qu'ils interfèrent entre eux, c'est-à-dire sans que les indices ne se mélangent. Cette signature permet également de pouvoir retrouver le critère qui est à l'origine d'un indice donné.

Terminons la description de la règle 7.3. Le masque *discriminant*

```
Discriminant : [P:cible.lexie]
```

génère la chaîne `2` pour chacune des six correspondances puisque la lexie de *détention* dans notre exemple est la lexie 2.

Comme le critère de notre exemple, *[lemme]-[ordonne]-[mot]*, n'est pas de la forme *[ems]-[<param2>]-[<param3>]*, la règle de la figure 7.4 n'est pas utilisée pour le modéliser. Pour finir cette modélisation, il ne reste plus qu'à ajouter la partie droite qui correspond à la règle de la figure 7.3. Exception de la clause *stop* que nous avons décrite, cette règle est la symétrique de la règle correspondant à la partie gauche. Elle génère les huit correspondances suivantes :

1. ***détention*** *illégale*
2. ***détention*** *illégale d*
3. ***détention*** *illégale d '*

4. ***détention*** *illégale d ' armes*
5. ***détention*** *illégale d ' armes de*
6. ***détention*** *illégale d ' armes de quatrième*
7. ***détention*** *illégale d ' armes de quatrième catégorie*
8. ***détention*** *illégale d ' armes de quatrième catégorie .*

Les chaînes générées pour ces huit correspondances par le masque référence sont donc respectivement :

1. `LOM_8_(1)_illégal`
2. `LOM_8_(2)_de`
3. `LOM_8_(3)_'`
4. `LOM_8_(4)_arme`
5. `LOM_8_(5)_de`
6. `LOM_8_(6)_quatrième`
7. `LOM_8_(7)_catégorie`
8. `LOM_8_(8)_.`

Finalement, le critère *[lemme]-[ordonne]-[mot]* appliqué sur notre extrait de corpus, génère dans le fichier d'apprentissage la ligne [2] suivante :

référence	*lexie*	*indices*	
M-10294-153	2	LOM_8_(-1)_de	LOM_8_(-2)_inculper
		LOM_8_(-3)_être	LOM_8_(-4)_avoir
		LOM_8_(-5)_boulanger	LOM_8_(-6)_le
		LOM_8_(1)_illégal	LOM_8_(2)_de
		LOM_8_(3)_'	LOM_8_(4)_arme
		LOM_8_(5)_de	LOM_8_(6)_quatrième
		LOM_8_(7)_catégorie	LOM_8_(8)_.

Évaluation des performances

Les tableaux 7.4 et 7.5 indiquent les meilleures précisions atteintes par les classifieurs TPCM(0,00) et TNB(0,00) pour les 24 critères, et les trois catégories grammaticales.

Dans la section 4.5.4, nous avons prévu que la désambiguïsation des verbes serait plus difficile que celle des adjectifs, qui serait elle-même plus difficile que celle des noms, en raison de l'entropie moyenne de la répartition des occurrences sur les lexies plus importante pour les verbes, intermédiaire pour les adjectifs et plus faible pour les noms. Effectivement, nous pouvons voir que la meilleure précision obtenue pour le classifieur TPCM(0,00) (respectivement pour le classifieur TNB(0,00)) est de 80,0% (81,9%) pour les noms, 75,2% (76,9%) pour les adjectifs et 69,0% (71,8%) pour les verbes. Cependant, cela ne veut pas dire que les deux classifieurs sont plus efficaces pour les noms que pour les adjectifs et les verbes. La mesure qui permet d'estimer leur efficacité est le gain et il est de 53,3% (57,6%) pour les noms, 53,8% (56,8%) pour les adjectifs et 50,6% (55,1%) pour les verbes. Ces chiffres montrent bien que l'efficacité des deux classifieurs et du critère le plus fiable (*[lemme]-[ordonne]-[mot]*) est comparable pour les trois catégories grammaticales.

2. Cette ligne est bien trop longue pour figurer sur cette page, nous avons donc effectué des retours à la ligne pour pouvoir la faire figurer dans son intégralité. Les en-têtes (*référence lexie indices*) précisent le sens des chaînes de caractères de la ligne. Bien entendu, ces en-têtes ne figurent pas dans le fichier d'apprentissage.

Nom du critère		Noms		Adjectifs		Verbes		**Moyennes**	
Complet	Abrev.	$P\%$	T	$P\%$	T	$P\%$	T	$P\%$	T
[jeton]- [ordonne] -[mot]	JOM	79,7	3	74,6	2	69,0	3	72,1	3
[lemme]- [ordonne] -[mot]	LOM	80,0	4	75,2	2	69,1	3	72,3	3
[ems]- [ordonne] -[mot]	EOM	68,8	2	61,4	1	55,6	2	59,3	2
[smallems]- [ordonne] -[mot]	SOM	64,4	1	58,9	1	51,3	2	55,1	2
[jeton]- [ordonne] -[mot-plein]	JOMp	78,4	1	72,1	2	61,0	2	66,4	2
[lemme]- [ordonne] -[mot-plein]	LOMp	79,1	1	72,3	1	61,3	2	66,8	2
[ems]- [ordonne] -[mot-plein]	EOMp	65,9	1	56,4	1	49,1	1	53,8	1
[smallems]- [ordonne] -[mot-plein]	SOMp	61,1	1	53,2	1	43,5	1	48,8	1
[jeton]-[non-ordonne]-[mot]	JNoM	76,9	2	73,5	1	66,7	2	69,8	2
[lemme]-[non-ordonne]-[mot]	LNoM	76,7	3	74,1	1	65,8	3	69,0	3
[ems]-[non-ordonne]-[mot]	ENoM	65,7	1	58,3	1	48,8	1	53,9	1
[smallems]-[non-ordonne]-[mot]	SNoM	61,2	1	55,6	1	45,7	1	50,6	1
[jeton]-[non-ordonne]-[mot-plein]	JNoMp	77,7	2	71,0	1	60,6	2	65,7	2
[lemme]-[non-ordonne]-[mot-plein]	LNoMp	77,6	1	71,0	1	59,9	2	65,2	2
[ems]-[non-ordonne]-[mot-plein]	ENoMp	63,9	1	53,9	1	45,7	1	50,8	1
[smallems]-[non-ordonne]-[mot-plein]	SNoMp	59,6	1	50,7	1	41,5	1	46,8	1
[jeton]- [differentie] -[mot]	JDM	78,7	2	74,3	1	68,3	3	71,1	2
[lemme]- [differentie] -[mot]	LDM	78,6	3	74,9	1	68,1	3	71,1	3
[ems]- [differentie] -[mot]	EDM	68,1	1	61,4	1	52,4	2	57,1	1
[smallems]- [differentie] -[mot]	SDM	64,3	1	58,9	1	49,1	1	53,9	1
[jeton]- [differentie] -[mot-plein]	JDMp	78,4	1	71,9	1	61,7	2	66,7	2
[lemme]- [differentie] -[mot-plein]	LDMp	79,1	1	72,3	1	61,8	2	66,9	2
[ems]- [differentie] -[mot-plein]	EDMp	65,9	1	56,4	1	49,1	1	53,8	1
[smallems]- [differentie] -[mot-plein]	SDMp	61,1	1	53,2	1	43,5	1	48,8	1
Performance optimale		80,0%	4	75,2%	2	69,1%	3	72,3%	3
Précision de l'algorithme majoritaire		57,3%		46,4%		37,2%		42,9%	

Tableau 7.4 – Performance du classifieur TPCM(0,00) pour les 24 critères et les trois catégories grammaticales. La colonne *Abrev.* contient le nom abrégé du critère. Dans la colonne $P\%$ est mentionnée la précision maximale atteinte par le critère pour une taille de contexte précisée dans la colonne T.

Nom du critère		Noms		Adjectifs		Verbes		**Moyennes**	
Complet	Abrev.	*P*%	T	*P*%	T	*P*%	T	*P*%	T
[jeton]- [ordonne] -[mot]	JOM	81,5	2	75,7	1	71,0	3	73,7	2
[lemme]- [ordonne] -[mot]	LOM	81,9	2	76,8	1	71,8	3	74,5	3
[ems]- [ordonne] -[mot]	EOM	73,4	2	63,9	2	63,0	2	65,4	2
[smallems]- [ordonne] -[mot]	SOM	68,1	2	60,8	1	57,3	2	60,0	2
[jeton]- [ordonne] -[mot-plein]	JOMp	78,9	1	71,6	1	59,5	1	65,6	1
[lemme]- [ordonne] -[mot-plein]	LOMp	79,0	1	72,1	1	60,0	1	66,0	1
[ems]- [ordonne] -[mot-plein]	EOMp	68,9	1	57,9	1	54,1	1	57,8	1
[smallems]- [ordonne] -[mot-plein]	SOMp	61,9	2	54,3	1	46,2	2	50,7	2
[jeton]-[non-ordonne]-[mot]	JNoM	79,5	2	74,6	1	69,5	2	72,3	2
[lemme]-[non-ordonne]-[mot]	LNoM	78,9	2	75,5	1	69,0	2	72,1	2
[ems]-[non-ordonne]-[mot]	ENoM	69,3	1	60,4	1	58,0	2	60,1	2
[smallems]-[non-ordonne]-[mot]	SNoM	63,6	1	58,1	1	50,7	2	54,3	1
[jeton]-[non-ordonne]-[mot-plein]	JNoMp	77,6	1	70,4	1	59,0	1	64,8	1
[lemme]-[non-ordonne]-[mot-plein]	LNoMp	77,0	1	70,6	1	58,7	1	64,5	1
[ems]-[non-ordonne]-[mot-plein]	ENoMp	64,5	1	54,6	1	50,5	1	54,1	1
[smallems]-[non-ordonne]-[mot-plein]	SNoMp	58,7	2	51,0	1	42,2	2	47,0	1
[jeton]- [differentie] -[mot]	JDM	81,0	2	75,7	1	70,6	2	73,5	2
[lemme]- [differentie] -[mot]	LDM	80,9	2	76,8	1	70,8	3	73,7	2
[ems]- [differentie] -[mot]	EDM	72,2	1	62,8	1	61,6	2	64,0	2
[smallems]- [differentie] -[mot]	SDM	66,9	1	60,8	1	55,4	2	58,3	2
[jeton]- [differentie] -[mot-plein]	JDMp	78,9	1	71,6	1	59,5	2	65,6	1
[lemme]- [differentie] -[mot-plein]	LDMp	79,0	1	72,1	1	60,0	1	66,0	1
[ems]- [differentie] -[mot-plein]	EDMp	68,9	1	57,9	1	54,1	1	57,8	1
[smallems]- [differentie] -[mot-plein]	SDMp	61,4	1	54,3	1	45,1	1	50,1	1
Performance optimale		81,9%	2	76,8%	1	71,8%	3	74,5%	3
Précision de l'algorithme majoritaire		57,3%		46,4%		37,2%		42,9%	

Tableau 7.5 – Performance du classifieur TNB(0,00) pour les 24 critères et les trois catégories grammaticales. Ce tableau se lit de la même manière que le tableau 7.4.

TPCM(0,00)				TNB(0,00)			
Noms	Adjectifs	Verbes	**Moyennes**	Noms	Adjectifs	Verbes	**Moyennes**
LOM	LOM	LOM	LOM	LOM	LOM	LOM	LOM
JOM	LDM	JOM	JOM	JOM	LDM	JOM	JOM
LOMp	JOM	JDM	JDM	JDM	JOM	LDM	LDM
LDMp	JDM	LDM	LDM	LDM	JDM	JDM	JDM
JDM	LNoM	JNoM	JNoM	JNoM	LNoM	JNoM	JNoM
LDM	JNoM	LNoM	LNoM	LOMp	JNoM	LNoM	LNoM
JOMp	LOMp	LDMp	LDMp	LDMp	LOMp	EOM	LOMp
JDMp	LDMp	JDMp	LOMp	LNoM	LDMp	EDM	LDMp
JNoMp	JOMp	LOMp	JDMp	JOMp	JOMp	LOMp	JOMp
LNoMp	JDMp	JOMp	JOMp	JDMp	JDMp	LDMp	JDMp
JNoM	LNoMp	JNoMp	JNoMp	JNoMp	LNoMp	JDMp	EOM
LNoM	JNoMp	LNoMp	LNoMp	LNoMp	JNoMp	JOMp	JNoMp
EOM	EOM	EOM	EOM	EOM	EOM	JNoMp	LNoMp
EDM	EDM	EDM	EDM	EDM	EDM	LNoMp	EDM
EOMp	SOM	SOM	SOM	ENoM	SOM	ENoM	ENoM
EDMp	SDM	EDMp	SDM	EOMp	SDM	SOM	SOM
ENoM	ENoM	EOMp	ENoM	EDMp	ENoM	SDM	SDM
SOM	EOMp	SDM	EDMp	SOM	SNoM	EOMp	EOMp
SDM	EDMp	ENoM	EOMp	SDM	EOMp	EDMp	EDMp
ENoMp	SNoM	SNoM	ENoMp	ENoMp	EDMp	SNoM	SNoM
SNoM	ENoMp	ENoMp	SNoM	SNoM	ENoMp	ENoMp	ENoMp
SOMp	SOMp	SOMp	SOMp	SOMp	SOMp	SOMp	SOMp
SDMp	SDMp	SDMp	SDMp	SDMp	SDMp	SDMp	SDMp
SNoMp	SNoMp	SNoMp	SNoMp	SNoMp	SNoMp	SNoMp	SNoMp

Tableau 7.6 – Les 24 critères classés par ordre de précision décroissante pour les trois catégories grammaticales et pour les deux classifieurs TPCM(0,00) et TNB(0,00).

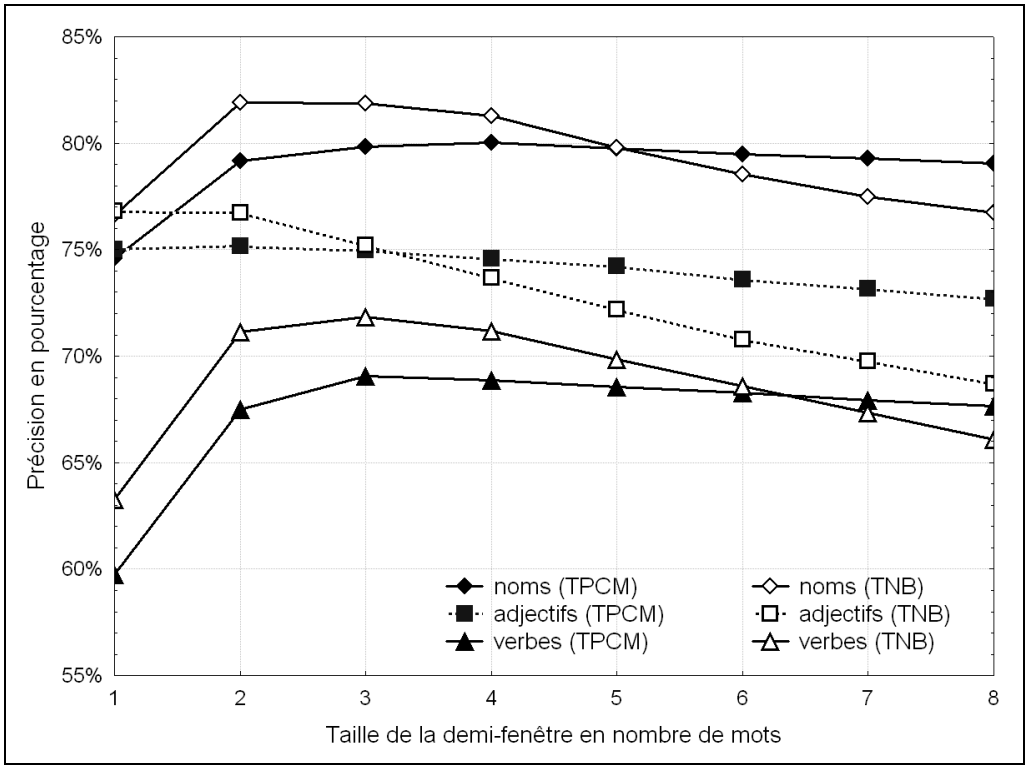


Figure 7.6 – Précision moyenne pour les trois catégories de vocables et les classifieurs TPCM(0,00) et TNB(0,00) du critère *[lemme]-[ordonne]-[mot]* en fonction de la taille du contexte.

Le tableau 7.6 montre les 24 critères classés par ordre de précision décroissante pour les trois catégories grammaticales et pour les deux classifieurs TPCM(0,00) et TNB(0,00). Ce tableau nous permet d'observer une certaine constance dans la précision des critères en fonction des catégories grammaticales et en fonction des classifieurs TPCM(0,00) et TNB(0,00). Ainsi, le critère le plus performant, *[lemme]-[ordonne]-[mot]* (*LOM* en abrégé), l'est pour chacune des catégories grammaticales et pour les deux classifieurs. La figure 7.6 détaille la précision moyenne atteinte par ce critère pour chacune des trois classes grammaticales de vocables et pour chacun des deux classifieurs en fonction de la taille du contexte.

Logiquement, le critère le moins performant est *[smallems]-[non-ordonne]-[mot-plein]* (*SNoMp* en abrégé). Ce critère ne s'intéresse qu'aux étiquettes morphosyntaxiques simplifiées des mots pleins sans distinction de position. Cela fait, au total, seulement huit valeurs d'indices distinctes (*ADJ*, *NCOM*, *NPRO*, *NHMIN*, *ADV*, *VCON*, *VINF*, *VPAR*). En se basant uniquement sur la présence de ces indices dans le contexte, la classification dépasse difficilement les performances de l'algorithme majoritaire.

Nous allons détailler et commenter les performances de ces critères dans les sections qui suivent.

7.3.5 Considérations sur la taille du contexte

Privilégier les petites tailles de contexte

Comme nous pouvons le voir dans les tableaux 7.4 et 7.5, les tailles de contextes avec lesquelles nous obtenons la meilleure précision sont petites et vont de plus ou moins un mot à plus ou moins quatre mots. Dans la majorité des cas, la meilleure précision est obtenue avec un contexte de plus ou moins un mot ou plus ou moins deux mots. Ces résultats sont en accord avec de nombreuses études, comme celles de Yarowsky (1993, 2000) et de El-Bèze *et al.* (1998) par exemple, qui obtiennent de bons résultats en ne se basant que sur de petits contextes.

Les classifieurs réagissent différemment à la variation de la taille du contexte

Les deux classifieurs que nous avons utilisés ne se comportent pas de la même façon vis-à-vis de la taille du contexte. La dynamique de la précision en fonction du contexte est plus grande pour le classifieur TNB(0,00) que pour le classifieur TPCM(0,00). Cette tendance est générale, la figure 7.7 le montre pour le critère *[lemme]-[ordonne]-[mot]*. Plus le contexte est grand, plus les indices sont bruités. Le classifieur TNB(0,00) qui combine l'information de tous les indices s'accommode moins bien de ce bruit que le classifieur TPCM(0,00) qui prend sa décision sur un indice unique supposé le plus fiable. En acceptant les mots en dehors de la phrase et en augmentant la taille du contexte, cette tendance est encore plus marquée comme le montre la figure 7.8

La prise en compte de la position des mots influe sur la décroissance de la précision lorsque la taille du contexte augmente

Un autre paramètre a un impact sur la dynamique de la précision en fonction du contexte, il s'agit du fait de tenir compte ou pas de la position des mots par rapport au mot à désambiguïser (*i.e.* considérer les mots du contexte comme un ensemble non ordonné ou ordonné). Le tableau 7.7 illustre ce phénomène pour le vocable *détention*. Nous utilisons ici le classifieur TPCM(0,00) qui ordonne les indices par fiabilité dé-

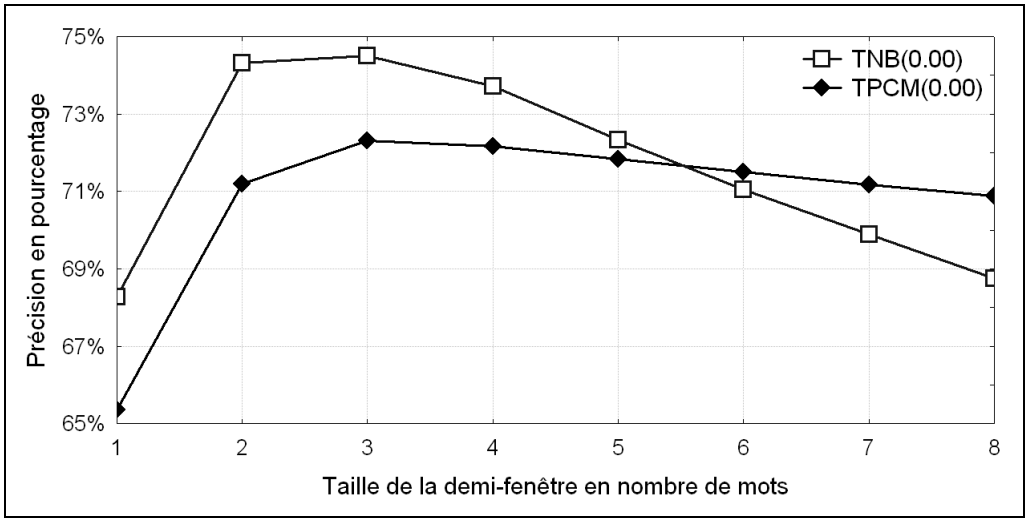


Figure 7.7 – Précision moyenne, pour les 60 vocables et pour le critère *[lemme]-[ordonne]-[mot]*, des classifieurs TPCM(0,00) et TNB(0,00).

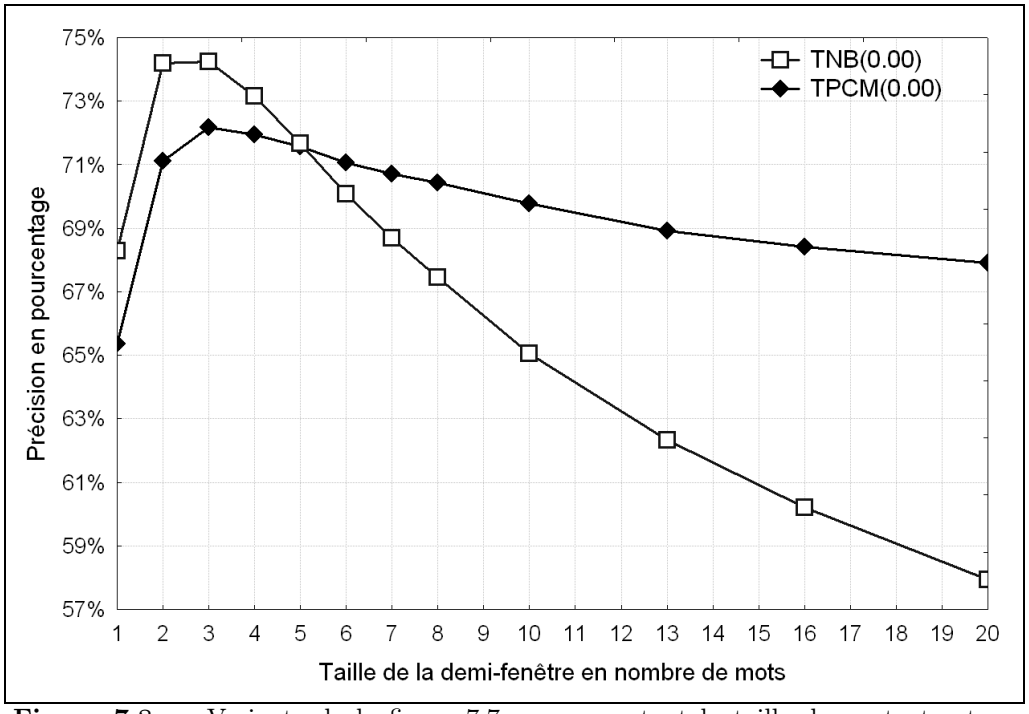


Figure 7.8 – Variante de la figure 7.7 en augmentant la taille du contexte et en acceptant les indices en dehors de la phrase.

LOM	1	2	fiabilité	LNoM	1	2	fiabilité
(-1)en	18	0	0,905	arme	0	16	0,900
(2)'	0	17	0,902	préventif	11	0	0,884
(3)arme	0	13	0,891	en	29	2	0,869
(1)préventif	11	0	0,884	'	22	31	0,575

Tableau 7.7 – Les indices pertinents pour le critère *[lemme]-[ordonne]-[mot]* (LOM) ne le sont pas forcément pour le critère *[lemme]-[non-ordonne]-[mot]* (LNoM). Les quatre colonnes de gauche montrent les quatre indices les plus fiables pour le critère *[lemme]-[ordonne]-[mot]* pour le vocable *détention*. Les quatre colonnes de droite montrent comment ces indices se fondent avec les autres mots du contexte avec le critère *[lemme]-[non-ordonne]-[mot]*. Dans les deux cas, la taille des contextes est de plus ou moins cinq mots à droite et à gauche. L'indice de fiabilité est celui calculé par le classifieur TPCM(0,00).

croissante (cf. section 6.5.4). L'intérêt de ce classifieur est que sa prise de décision est transparente à travers sa liste de décisions.

Les deux lexies de *détention* (cf. annexe C.4) sont :

1. « Fait d'être incarcéré ou enfermé » ;
2. « Avoir en sa possession ».

Selon le tableau 7.7, le mot *en* adjacent à gauche est l'indice le plus fiable pour le critère *[lemme]-[ordonne]-[mot]*. C'est un bon indicateur de la lexie 1 comme le montrent les exemples suivants :

– « ... maintenir des personnes **en détention** sans inculpation ni jugement ... » ;
– « ... avait été mis **en détention** préventive et relâché après ... » ;
– « Aux CRS , la " présence dissuasive **en détention** " .. ».

L'apostrophe en deuxième position à droite est le deuxième indice le plus fiable pour le critère *[lemme]-[ordonne]-[mot]*. C'est un bon indicateur de la lexie 2 comme le montrent les exemples suivants :

– « ... relative au contrôle de l ' acquisition et de la **détention** d **'** armes . » ;
– « ... **détention** d **'** une arme à feu pendant un voyage ... » ;
– « ... aboutir à la **détention** d **'** un diplôme . ».

Comme le montre le tableau 7.7, ces deux indices les plus fiables, pour le critère *[lemme]-[ordonne]-[mot]*, ont une mesure de fiabilité de 0,905 et 0,902. En raison du bruit généré par des mots se trouvant dans d'autres positions, cette fiabilité tombe respectivement à 0,869 et à 0,575 avec le critère *[lemme]-[non-ordonne]-[mot]*. La figure 7.9 illustre également ce comportement. Quand le contexte croît, la pente de la décroissance de la précision est plus importante pour le critère *[lemme]-[non-ordonne]-[mot]* que pour le critère *[lemme]-[ordonne]-[mot]*.

Le tableau 7.7 peut laisser penser que ce phénomène ne se produit que pour des mots grammaticaux et non pour des mots pleins. En effet, l'indice *(1)préventif* n'est pas affecté par le passage au critère *[lemme]-[non-ordonne]-[mot]*, et l'indice *(3)arme* est encore plus pertinent pour le critère *[lemme]-[non-ordonne]-[mot]*. La figure 7.10 montre que ce comportement reste vrai même quand le critère ne considère que les mots pleins.

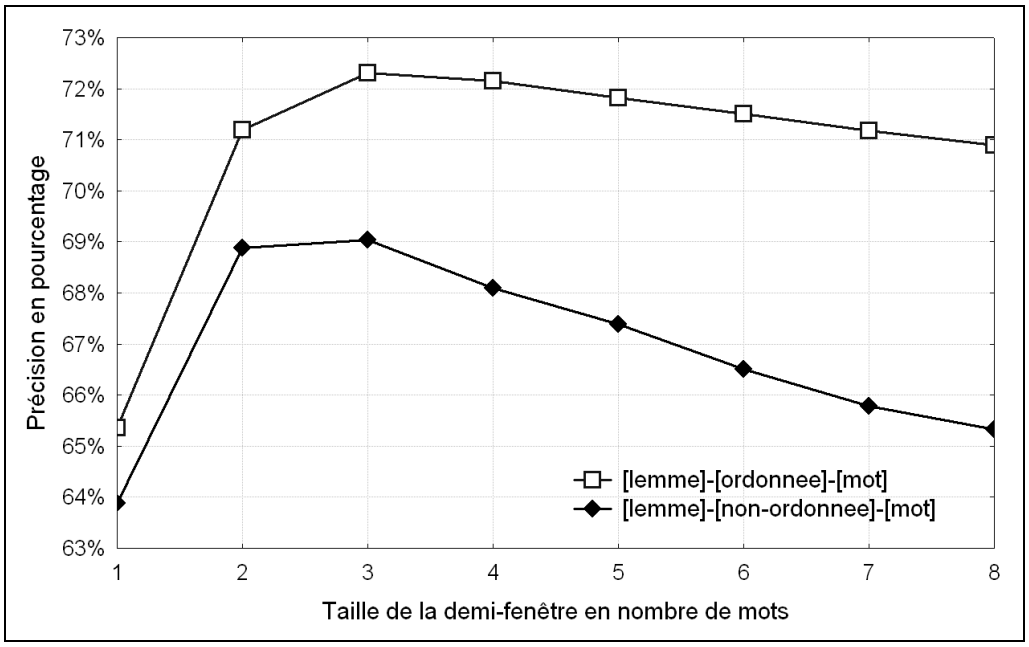


Figure 7.9 – Précision moyenne du classifieur TPCM(0,00), pour les 60 vocables, et pour les critères *[lemme]-[ordonne]-[mot]* et *[lemme]-[non-ordonne]-[mot]*.

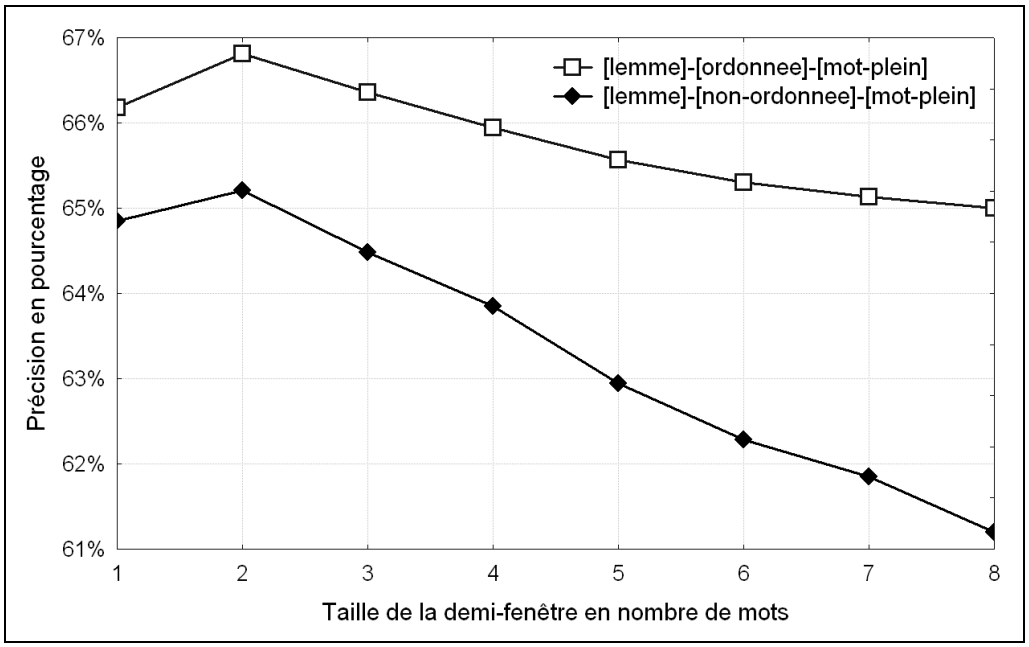


Figure 7.10 – Précision moyenne du classifieur TPCM(0,00), pour les 60 vocables, et pour les critères *[lemme]-[ordonne]-[mot-plein]* et *[lemme]-[non-ordonne]-[mot-plein]*.

La taille optimale du contexte dépend de la catégorie grammaticales du mot à désambiguïser

Intéressons-nous aux performances des critères de la forme *[jeton]-[<param2>]-[<param3>]* ou *[lemme]-[<param2>]-[<param3>]* dans les tableaux 7.4 et 7.5. Dans ces tableaux, la taille optimale du contexte est pratiquement systématiquement plus grande pour les critères qui considèrent tous les mots que pour les critères qui ne considèrent que les mots pleins, exception faite des adjectifs. Il est évident que les mots pleins apportent une information essentielle. Aussi, pour être performant, un contexte doit en contenir. Or, contrairement aux noms et aux verbes, les adjectifs ont, dans la majorité des cas, un mot plein directement adjacent. Ainsi, le meilleur contexte pour désambiguïser un adjectif est généralement de plus ou moins un mot, que le critère s'intéresse à tous les mots ou uniquement aux mots pleins, tandis que pour les noms et les verbes, le meilleur contexte est de plus ou moins un à plus ou moins deux mots quand le critère ne considère que les mots pleins, et de plus ou moins deux à plus ou moins quatre mots quand il considère tous les mots. La figure 7.6 illustre bien ce comportement des trois catégories de vocables. Cette figure permet également de noter que :

- bien que le meilleur contexte pour les adjectifs soit de plus ou moins un mot, en prenant un contexte plus grand, la dégradation de la précision est pratiquement nulle pour un contexte de plus ou moins deux ou plus ou moins trois mots pour le classifieur TPCM(0,00), pratiquement nulle pour un contexte de plus ou moins deux et de l'ordre de 1,6% pour un contexte de plus ou moins trois mots pour le classifieur TNB(0,00) ;
- le meilleur contexte pour les noms est de plus ou moins deux ou plus ou moins quatre mots suivant le classifieur utilisé ; en revanche, en prenant un contexte plus petit, la dégradation de la précision est plus importante que pour les adjectifs et est l'ordre de 5,4% pour un contexte de plus ou moins un mot ;
- le meilleur contexte pour les verbes est de plus ou moins trois mots ; en prenant un contexte plus petit, la dégradation de la précision est encore plus importante, elle est l'ordre de 9% pour un contexte de plus ou moins un mot.

Les catégories grammaticales réagissent différemment à un élargissement de la taille du contexte

Lorsque nous sortons du cadre des micro-contextes et que nous nous intéressons à des contextes plus larges, nous remarquons que l'information pour désambiguïser un mot semble plus locale quand ce mot est un verbe ou un adjectif que quand il s'agit d'un nom. Yarowsky (1993) a déjà observé ce phénomène. Pour le mettre en valeur, nous avons tenté de désambiguïser chacun des 60 vocables en regardant le lemme de quatre mots pleins situés à une distance de plus ou moins x mots (la distance est bien comptée en nombre de mots et non en nombre de mots pleins). Nous avons effectué cette expérience avec les deux classifieurs et obtenu des résultats similaires. Nous exposons les résultats obtenus avec le classifieur TPCM(0,00). Pour effectuer la désambiguïsation de chacun des 60 vocables, ce classifieur dispose donc de huit indices :

- le lemme des quatre mots pleins les plus proches situés à une distance de plus de x mots à gauche de la cible ;
- le lemme des quatre mots pleins les plus proches situés à une distance de plus de x mots à droite de la cible.

La figure 7.11 montre le gain moyen obtenu, en fonction de cette distance de x mots, pour les 20 noms, les 20 adjectifs et les 20 verbes. Nous observons immédiatement

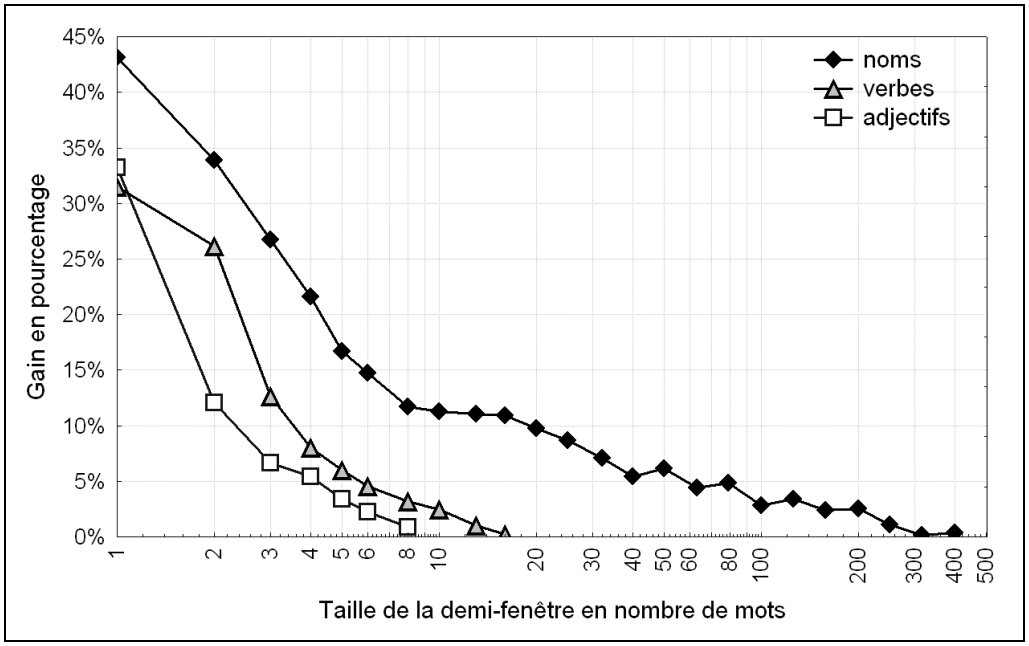


Figure 7.11 – Gain moyen pour chacune des catégories en fonction de l'éloignement des indices. Il s'agit du gain obtenu par le classifieur TPCM(0,00) en basant sa classification sur le lemme de quatre mots pleins situés à une distance de plus ou moins x mots (représentée par une échelle logarithmique sur la figure).

qu'en s'éloignant de la cible, le gain tend vers zéro de manière bien plus rapide pour les adjectifs et les verbes que pour les noms. Le gain devient nul au delà de 8 mots pour les adjectifs, 14 pour les verbes et 300 pour les noms.

7.3.6 Faut-il favoriser la variabilité ou le regroupement ?

L'apprentissage de nos deux classifieurs est basé sur la corrélation qu'il peut y avoir entre la présence d'un indice (*i.e.* la valeur d'un attribut) et la lexie de l'exemple. Pour cela, il faut que différents exemples aient des indices identiques. Lorsque la variabilité des attributs est trop grande, chaque description d'exemple est différente, la généralisation et donc l'apprentissage sont alors impossibles. Ce phénomène, discuté dans la section 2.5.6, est d'autant plus sensible que le nombre d'exemples d'apprentissage est petit.

Il en va de même si la variabilité des indices est trop faible. Dans ce cas, les attributs ont des valeurs quasiment constantes, aucune corrélation entre la présence d'un indice et la lexie de l'exemple ne peut être faite, l'apprentissage est alors également impossible.

Le tableau 7.8 indique le nombre d'indices générés par nos 24 critères pour un contexte de plus ou moins deux mots et pour l'ensemble des 60 vocables. En première lecture, il peut paraître surprenant que le critère *JOMp* génère plus d'indices que le critère *JOM*. En effet, la seule différence entre ces deux critères est que *JOMp* ne considère que les mots pleins alors que *JOM* considère tous les mots, or l'ensemble des mots pleins est inclus dans l'ensemble des mots. L'inversion qu'il se produit avec le dénombrement des indices provient du fait que le contexte est limité à plus ou moins deux mots. Dans ce contexte, seul un sous-ensemble des mots pleins existant est concerné. Ce sous-ensemble est encore réduit dans le cas où le critère considère également les mots

Critères	Nb. Ind.	Critères	Nb. Ind.	Critères	Nb. Ind.
JOMp	53 663	LNoMp	17 586	ENoM	125
JDMp	40 667	JNoM	17 060	ENoMp	93
LOMp	35 030	LDM	14 770	SOM	84
JNoMp	30 065	LNoM	10 901	SDM	44
JOM	26 328	EOM	498	SOMp	32
LDMp	25 036	EOMp	418	SNoM	22
JDM	21 748	EDM	301	SDMp	16
LOM	18 744	EDMp	255	SNoMp	8

Tableau 7.8 – Critères classés par nombre décroissant d'indices générés. Dans cette expérience, la taille du contexte est de plus ou moins deux mots. La colonne *Nb. Ind.* précise le nombre d'indices générés pour l'ensemble des 60 vocables.

grammaticaux. Or, la variabilité des mots pleins est bien plus importante que celle des mots grammaticaux. C'est cette faible variabilité des mots grammaticaux qui explique pourquoi les critères qui considèrent tous les mots génèrent moins d'indices que ceux qui ne considèrent que les mots pleins dans le tableau. D'ailleurs, cela n'est vrai que pour les critères qui considèrent le *lemme* ou le *jeton* des mots, dans le cas où c'est l'étiquette *ems* ou *smallems* qui est considérée, l'ordre logique est rétabli.

En excluant les critères qui ne considèrent que les mots pleins, le tableau 7.9 montre la corrélation qu'il y a entre le nombre d'indices générés (la variabilité) et la précision de la classification. Pour ces critères, il semble que variabilité croissante et précision croissante aillent de pair.

Le tableau 7.10 montre l'impact sur la précision du choix de chacune des trois parties *[<param1>]*, *[<param2>]* et *[<param3>]* des 24 critères étudiés avec le classifieur TPCM(0,00). Le tableau 7.11 montre cet impact en utilisant le classifieur TNB(0,00).

Ces tableaux permettent de tirer un nombre important d'enseignements :

- En utilisant le classifieur TPCM(0,00) (respectivement TNB(0,00)), considérer tous les mots plutôt que seulement les mots pleins améliore en moyenne la précision de 0,3% (2,2%) pour les noms, 2,5% (4,1%) pour les adjectifs et 6,9% (11,1%) pour les verbes. Il semble donc que les mots grammaticaux soient très importants pour la désambiguïsation des verbes, moins importants pour la désambiguïsation des adjectifs et peu importants pour la désambiguïsation des noms. Comme nous l'avons noté dans la section 7.2.1, de nombreuses études ne considèrent pas les mots grammaticaux (les mots vides). Dans notre expérience, si une telle pratique ne semble pas trop préjudiciable pour les noms, elle l'est pour les adjectifs et encore plus pour les verbes.
- La lemmatisation ne s'avère pas constituer une opération essentielle car les critères de la forme *[lemme]-[<param2>]-[<param3>]* ne se démarquent pas franchement des critères de la forme *[jeton]-[<param2>]-[<param3>]*.
- Nous observons que le classifieur TNB(0,00) réagit mieux que le classifieur TPCM(0,00) lorsque les critères considèrent tous les mots (plutôt que seulement les mots pleins) et lorsque les critères considèrent les étiquettes morphosyntaxiques (*ems*) des mots.

Conformément au tableau 7.9, l'analyse des tableaux 7.10 et 7.11 va dans le sens de la variabilité. Par exemple, les critères dans lesquels *<param2>=ordonne* génèrent plus d'indices que les critères similaires dans lesquels *<param2>=differencie* qui génèrent eux-mêmes plus d'indices que les critères dans lesquels *<param2>=non-ordonne*. La

Critères	TPCM(0,00)	Critères	TNB(0,00)	Critères	Nb. Ind.
LOM	71,2%	LOM	74,3%	**JOM**	26 328
JOM	71,5%	LDM	73,6%	**JDM**	21 748
JDM	71,1%	JOM	73,6%	LOM	18 744
LDM	70,6%	JDM	73,5%	**JNoM**	17 060
JNoM	69,7%	JNoM	72,2%	**LDM**	14 770
LNoM	68,8%	*LNoM*	72,1%	***LNoM***	10 901
EOM	59,3%	*EOM*	65,5%	***EOM***	498
EDM	56,5%	*EDM*	64,1%	***EDM***	301
SOM	55,1%	SOM	60,2%	**ENoM**	125
ENoM	52,9%	ENoM	60,0%	**SOM**	84
SDM	52,1%	*SDM*	58,5%	***SDM***	44
SNoM	49,2%	*SNoM*	53,7%	***SNoM***	22

Tableau 7.9 – Correspondance entre la variabilité et la précision. Dans les colonnes un, trois et cinq, les critères en italiques sont ceux qui se trouvent sur la même ligne dans ces trois colonnes. Dans la cinquième colonne, les critères en gras sont ceux qui se trouvent sur la même ligne ou sur la ligne au-dessous ou au-dessus dans la première ou la troisième colonne. Les deuxième, quatrième et sixième colonnes indiquent respectivement la précision obtenue par le classifieur TPCM(0,00), la précision obtenue par le classifieur TNB(0,00) et le nombre d'indices générés. Dans tous les cas, la taille du contexte est de plus ou moins deux mots.

	Noms	Adjectifs	Verbs	Moy
[jeton]-	16,3%	17,8%	18,8%	18,0%
[lemme]-	16,6%	18,2%	18,5%	17,9%
[ems]-	4,4%	2,9%	4,4%	4,1%
[smallems]-	0,0%	0,0%	0,0%	0,0%
-[ordonne]-	2,9%	1,1%	2,2%	2,3%
-[differencie]-	1,9%	0,8%	1,6%	1,3%
-[non-ordonne]-	0,0%	0,0%	0,0%	0,0%
-[mot]	0,3%	2,5%	6,9%	4,7%
-[mot-plein]	0,0%	0,0%	0,0%	0,0%

Tableau 7.10 – Évaluation de l'impact du choix de chacune des trois parties *[<param1>]*, *[<param2>]* et *[<param3>]* des 24 critères étudiés avec le classifieur TPCM(0,00). Le tableau se lit de la manière suivante : la valeur de *<param1>* qui donne les plus mauvais résultats est *<param1>=smallems*, en prenant *<param1>=ems*, la précision est en moyenne améliorée de 4,1% et de 17,9% en prenant *<param1>=lemme*. Les précisions retenues pour calculer les moyennes sont celles obtenues, pour chacun des critères, avec la taille de contexte optimal.

	Noms	**Adjectifs**	**Verbs**	**Moy**
[jeton]-	16,1%	16,7%	15,4%	15,8%
[lemme]-	16,0%	17,5%	15,5%	16,0%
[ems]-	6,1%	3,0%	7,4%	6,5%
[smallems]-	0,0%	0,0%	0,0%	0,0%
-[ordonne]-	2,0%	1,0%	1,5%	1,3%
-[differencie]-	1,5%	1,0%	1,1%	1,1%
-[non-ordonne]-	0,0%	0,0%	0,0%	0,0%
-[mot]	2,2%	4,1%	11,1%	7,9%
-[mot-plein]	0,0%	0,0%	0,0%	0,0%

Tableau 7.11 – Évaluation de l'impact du choix de chacune des trois parties *[<param1>]*, *[<param2>]* et *[<param3>]* des 24 critères étudiés avec le classifieur TNB(0,00). Le tableau se lit de la même manière que le tableau 7.10.

précision croissante va, dans la plupart des cas, dans le même sens que le nombre d'indices générés. Ce résultat général est à moduler dans deux cas :

- un critère dans lequel *<param3>=mot* génère potentiellement plus d'indices qu'un critère similaire pour lequel *<param3>=mot-plein*, cependant, comme nous l'avons vu plus haut, cette affirmation n'est pas vraie pour de petites tailles de contexte ;
- un critère dans lequel *<param1>=jeton* génère plus d'indices qu'un critère similaire dans lequel *<param1>=lemme*, cependant, nous n'observons pas ou peu d'amélioration de la précision dans le cas où *<param1>=jeton*.

7.3.7 Considérations sur les indices

Dans cette section, nous cherchons à observer les indices qui permettent la levée de l'ambiguïté. Plus précisément, nous nous intéressons à la catégorie grammaticale de ces indices et à leur répartition spatiale autour du mot à désambiguïser. Pour mener à bien cette étude, nous utilisons le classifieur TPCM(0,00) en raison de sa prise de décision via une liste de décisions qui sélectionne un indice unique pour effectuer la classification. La transparence de cette prise de décision nous permet d'observer la catégorie grammaticale et la répartition spatiale de cet indice unique sélectionné pour la prise de décision par le classifieur TPCM(0,00).

Nous nous intéressons à la catégorie grammaticale indiquée par l'étiquette *smallems*, la granularité des étiquettes *ems* étant trop fine pour cette étude. En toute logique nous devrions utiliser le critère le plus performant : *[lemme]-[ordonne]-[mot]*. Cependant, la granularité des étiquettes *smallems* est suffisamment fine pour contenir trois étiquettes distinctes pour les verbes (*VCON*, *VINF* et *VPAR*). Or, cette information n'est pas contenue dans l'étiquette *lemme*. Nous utilisons donc le critère *[jeton]-[ordonne]-[mot]* qui permet d'extraire suffisamment d'informations plutôt que le critère *[lemme]-[ordonne]-[mot]* qui obtient les meilleures performances. De toute façon, la différence de précision moyenne entre ces deux critères n'est que de 0,2% pour le classifieur TPCM(0,00) (cf. tableau 7.4). Le contexte retenu est de plus ou moins trois mots puisque c'est celui qui donne les meilleurs résultats en moyenne (cf. tableau 7.4).

Le tableau 7.12 synthétise les résultats de cette étude. Il permet de tirer les enseignement suivants (nous citons, entre parenthèses et dans l'ordre, les résultats obtenus pour les noms, les adjectifs puis les verbes) :

- les noms communs obtiennent l'une des meilleures précisions (93,0% ; 93,7% ;

smallems	**Noms**			**Adjectifs**			**Verbes**		
des indices	Pré.%	Uti.%	Pro.%	Pré.%	Uti.%	Pro.%	Pré.%	Uti.%	Pro.%
NCOM	93,0	12,74	16,09	93,7	25,33	23,35	87,8	25,96	19,44
DET	73,8	30,19	23,86	69,8	21,33	17,63	48,1	12,65	15,98
PREP	78,3	24,17	21,16	61,4	15,20	14,88	62,9	17,39	16,26
ADJ	94,9	13,65	10,28	80,3	2,16	6,27	65,4	3,24	5,90
PCTFAIB	71,8	6,07	6,25	63,9	7,01	7,66	57,2	4,01	6,17
ADV	57,0	1,07	2,64	79,2	9,56	6,22	60,3	5,67	6,01
PROPE	63,6	0,58	1,49	67,0	2,61	2,54	65,9	11,38	9,16
PCTFORTE	72,1	3,00	2,44	69,0	4,16	3,37	80,9	2,90	2,07
COO	72,0	2,96	2,92	65,9	4,49	3,68	40,8	1,21	2,10
VCON	67,9	1,45	4,64	53,9	2,37	5,73	54,4	3,01	5,11
SUB	78,6	0,62	1,08	58,1	1,84	2,12	79,9	2,73	1,88
VINF	90,2	0,90	1,66	80,6	0,71	1,57	87,2	2,92	2,21
PRORE	58,6	0,77	1,20	40,6	0,74	1,21	61,2	2,52	2,53
Résidu	72,8	1,00	0,00	54,9	1,05	0,00	54,2	0,92	0,00
NPRO	86,8	0,33	1,44	92,0	0,58	1,17	81,8	1,22	2,50
VPAR	89,7	0,34	2,42	50,0	0,16	1,70	81,0	0,92	1,31
PRODE	100,0	0,04	0,21	35,0	0,23	0,56	68,6	0,77	0,67
PROIN	25,0	0,04	0,19	64,7	0,20	0,25	51,8	0,33	0,40
AlgoMAJ	50,0	0,11	·	47,1	0,20	·	10,0	0,03	·
NHMIN	·	·	0,01	57,1	0,08	0,03	·	·	0,00
PROPO	·	·	0,00	·	·	0,00	100,0	0,01	0,01
INT	·	·	0,01	·	·	0,00	·	·	0,01
TOTAL	79,8	11 359	63 733	74,1	8686	48 411	69,0	33 751	190 828
MAJ	57,3			46,4			37,2		

Tableau 7.12 – Précision, utilisation et proportion en fonction de l'étiquette morphosyntaxique simplifiée *smallems* des indices utilisés par le classifieur TPCM(0,00) pour le critère *[jeton]-[ordonne]-[mot]*, en considérant un contexte de plus ou moins trois mots. La colonne *Pré.%* donne la précision en pourcentage de la désambiguïsation réalisée en utilisant les indices dont l'étiquette *smallems* est précisée dans la première colonne. La colonne *Uti.%* précise en pourcentage le nombre d'indices d'étiquette *smallems* (première colonne) utilisés pour l'ensemble des vocables désambiguïsés. La colonne *Pro.%* indique la proportion en pourcentage des indices d'étiquette *smallems* présents dans le contexte des vocables. La ligne *Résidu* correspond à des indices n'ayant pas reçu d'étiquette morphosyntaxique. La ligne *AlgoMAJ* correspond à des vocables qui n'ont pas reçu d'étiquette lexicale par le classifieur TPCM(0,00) et qui ont par conséquent été étiquetés par l'étiquette lexicale majoritaire. La ligne *TOTAL* donne la précision moyenne, le nombre de d'indices utilisés (qui correspond au nombre de vocables désambiguïsés) et le nombre d'indices présents dans le contexte (le contexte de chaque vocable peut contenir jusqu'à six indices bien qu'un seul soit utilisé pour la désambiguïsation). La ligne *MAJ* rappelle enfin la précision de l'algorithme majoritaire.

87,8%) et constituent l'une des catégories d'indices la plus utilisée (12,7% ; 25,3% ; 26%) pour les trois catégories de vocables ;
- les adjectifs constituent de bons indices pour les noms (précision de 94,9%) et pour les adjectifs (précision de 80,3%) mais sont surtout utiles aux noms puisqu'ils sont utilisés dans 13,7% des cas contre seulement 2,2% pour les adjectifs ;
- les adverbes sont essentiellement intéressants pour les adjectifs pour lesquels la précision atteint 79,2% avec une utilisation dans 9,6% des cas ;
- les verbes à l'infinitif constituent des indices très fiables pour les trois catégories de vocables (précision de 90,2% ; 80,6% ; 87,2%), mais ils sont peu utilisés (0,9% ; 0,7% ; 2,9%) car peu présents dans le contexte (1,7% ; 1,6% ; 2,2%) ;
- les conjonctions de subordination sont de bons indicateurs pour les noms (78,6%) et pour les verbes (79,9%) mais sont peu utilisées (0,6% ; 2,7%) car peu présentes dans le contexte (1,1% ; 1,9%) ;
- il est intéressant de remarquer que les indices les plus pertinents sont privilégiés par le classifieur TPCM(0,00) ; ainsi les adjectifs représentent 10,3% des indices du contexte des noms mais sont utilisés dans 13,7% des cas, les noms communs représentent 23,4% des indices du contexte des adjectifs et 19,4% du contexte des verbes mais sont utilisés dans 25,3% des cas pour les adjectifs et 26% des cas pour les verbes ;
- la faible précision des verbes conjugués (67,9% ; 53,9% ; 54,4%) est à noter ;
- les catégories grammaticales qui obtiennent la meilleure précision sur l'ensemble des 60 vocables sont les noms (communs et propres), les verbes à l'infinitif et au participe, les adjectifs et les conjonctions de subordination.

L'idée sous-jacente à l'utilisation de critères qui ne considèrent que les mots pleins (par exemple *[jeton]-[ordonne]-[mot-plein]*) est que ces mots constituent des indices plus fiables que les autres. Il s'agit d'une conjecture *a priori*. Le tableau 7.12 nous renseigne sur la précision de la désambiguïsation en fonction de l'étiquette morphosyntaxique simplifiée (*i.e.* étiquette *smallems*) des indices. Il est donc tentant de se servir de cette information pour évaluer les performances d'un critère construit en suivant la même idée que le critère *[jeton]-[ordonne]-[mot-plein]* mais en n'utilisant que les catégories d'indices pour lesquelles la précision de la désambiguïsation observée (*a posteriori*) est importante. Nous définissons ainsi un nouveau critère (*[jeton]-[ordonne]-[mot-select]*) qui ne considère que les catégories d'indices qui obtiennent une bonne précision comparée à la précision moyenne obtenue pour le classifieur *[jeton]-[ordonne]-[mot]*. Nous retenons les catégories d'indices suivantes :
- pour les noms, les indices dont l'étiquette *smallems* est *NCOM*, *PREP*, *ADJ*, *SUB*, *VINF*, *NPRO*, *VPAR* ou *PRODE* ;
- pour les adjectifs, les indices dont l'étiquette *smallems* est *NCOM*, *DET*, *ADJ*, *ADV*, *VINF* ou *NPRO* ;
- pour les verbes, les indices dont l'étiquette *smallems* est *NCOM*, *ADJ*, *PROPE*, *PCTFORTE*, *SUB*, *VINF*, *NPRO*, *VPAR* ou *PRODE*.

Le tableau 7.13 présente les résultats de cette expérience. La colonne $P\%$ indique la précision en pourcentage réalisée sur les mots étiquetés par les classifieurs PCM(0,00) et NB(0,00). Ces classifieurs ne réalisent pas une classification sur toutes les occurrences. Le critère *[jeton]-[ordonne]-[mot]* est celui dont la variabilité des indices générés est la plus faible et donc celui qui permet d'étiqueter un maximum d'occurrences. Le critère *[jeton]-[ordonne]-[mot-plein]* est celui dont la variabilité des indices générés est la plus grande et donc celui avec lequel le moins d'occurrences sont étiquetées. Les deux colonnes sur lesquelles les critères peuvent être comparés sont les colonnes $PG\%$

et *PT*%. Le critère *[jeton]-[ordonne]-[mot]* reste le plus performant. Les performances du critère *[jeton]-[ordonne]-[mot-select]* sont comparables à celles du critère *[jeton]-[ordonne]-[mot-plein]*. Les indices générés par le critère *[jeton]-[ordonne]-[mot]* sont peu bruités puisque le contexte est extrêmement réduit (de un à trois mots) et qu'en outre les mots sont différenciés par leur position. Ainsi, quand le classifieur TPCM(0,00) base sa classification sur une classe d'indices pour laquelle la précision est modeste, c'est probablement parce qu'il n'y a pas de meilleur indice disponible dans le contexte (s'il y en avait un, ce serait lui qui serait utilisé). Dans un tel contexte peu bruité, il est probablement préférable de laisser le classifieur TPCM(0,00) trier et sélectionner le meilleur indice plutôt, que d'effectuer un pré-filtrage des indices peu fiables puisque, dans certains cas, ces indices sont certainement les plus pertinents. Le fait de filtrer les indices amène également à aller chercher de l'information à une distance du mot à désambiguïser un peu plus grande. Une conjecture pour expliquer que le critère *[jeton]-[ordonne]-[mot]* reste le plus performant provient peut-être de cet éloignement qui dégrade la pertinence des indices.

Les figures 7.12, 7.13 et 7.14 montrent la répartition spatiale des principales catégories grammaticales des indices utilisés pour lever l'ambiguïté de chacune de nos trois catégories de vocables avec le classifieur TPCM(0,00), le critère *[jeton]-[ordonne]-[mot]* et un contexte de plus ou moins trois mots.

Dans la section 7.3.5 page 182, nous avons remarqué une dégradation importante de la précision pour les verbes, et dans une moindre mesure pour les noms, lorsque le contexte n'était que de plus ou moins un mot en utilisant le critère *[lemme]-[ordonne]-[mot]*. Les trois figures que nous venons de citer, bien qu'elles soient réalisées en utilisant un autre critère (*[jeton]-[ordonne]-[mot]*) permettent d'expliquer ce phénomène comme nous allons le voir dans les paragraphes qui suivent.

L'information la plus pertinente pour les adjectifs est véhiculée par les noms communs et les adverbes. La figure 7.13 montre que les noms et les adverbes sont très majoritairement adjacents aux adjectifs, et répartis de manière égale en position ± 1 pour les noms, et exclusivement en position -1 pour les adverbes. De plus, ce contexte de plus ou moins un mot contient 54% des indices utilisés pour la désambiguïsation des adjectifs. Nous comprenons maintenant très bien pourquoi les adjectifs se satisfont d'un contexte de plus ou moins un mot.

L'information la plus pertinente pour les noms est véhiculée par les adjectifs et les noms communs. La figure 7.12 montre que les adjectifs sont adjacents aux noms, et se situent majoritairement en position $+1$. Les noms communs se situent eux majoritairement en position -2 et $+2$. Ainsi, un contexte réduit à plus ou moins un mot perd toute l'information qui peut être véhiculée par les noms communs du contextes, information qui est pourtant de première importance. Ce contexte de plus ou moins un mot contient tout de même 57% des indices utilisés pour la désambiguïsation des noms. Les positions -2 et $+2$ contiennent 22% des indices utilisés pour la désambiguïsation des noms. Ce même contexte n'en contient que 19% pour les adjectifs. La désambiguïsation des noms nécessite un contexte plus large que le micro-contexte de plus ou moins un mot. Comme nous l'avons déjà dit, les prépositions (*PREP*) sont utiles à la désambiguïsation des noms et sont bien réparties autour de celui-ci, avec une concentration accentuée en position -1 et $+1$. Logiquement, les déterminants (*DET*) sont essentiellement positionnés en position -1.

L'information la plus pertinente pour les verbes est véhiculée par les noms communs. La figure 7.14 montre que ces noms communs sont très majoritairement situés en position $+2$ et, dans une moindre mesure, en position $+3$. Le contexte de plus ou moins un mot contient seulement 37% des indices utilisés pour la désambiguïsation des

		Noms				Adjectifs				Verbes			
		P%	*PG%*	*PT%*	T	*P%*	*PG%*	*PT%*	T	*P%*	*PG%*	*PT%*	T
PCM	[jeton]-[ordonne]-[mot]	79,8	63,4	79,7	3	74,7	61,7	74,6	2	69,0	58,4	69,0	3
	[jeton]-[ordonne]-[mot-plein]	86,9	64,8	78,4	1	78,8	61,1	72,1	2	65,7	50,7	61,0	2
	[jeton]-[ordonne]-[mot-select]	79,8	62,1	78,5	2	76,9	59,3	70,9	1	71,1	57,5	66,1	1
NB	[jeton]-[ordonne]-[mot]	81,6	67,0	81,5	2	76,5	63,7	75,7	1	71,0	61,3	71,0	3
	[jeton]-[ordonne]-[mot-plein]	87,7	65,9	78,9	1	81,0	60,7	71,6	1	67,8	50,8	59,5	1
	[jeton]-[ordonne]-[mot-select]	80,6	63,6	79,2	2	77,6	60,2	71,5	1	71,3	57,8	66,3	1

Tableau 7.13 – Performances moyennes, pour les trois catégories de vocables, des classifieurs TPCM(0,00) et TNB(0,00) pour un critère où les indices sont sélectionnés (*[jeton]-[ordonne]-[mot-select]*). Les performances de deux autres critères (*[jeton]-[ordonne]-[mot]* et *[jeton]-[ordonne]-[mot-plein]*) sont données à titre de comparaison. Les performances de ces deux critères ont déjà partiellement été données dans les tableaux 7.4 et 7.5. La colonne $P\%$ indique la précision en pourcentage réalisée sur les mots étiquetés par les classifieurs (*i.e.* précision des classifieurs PCM(0,00) et NB(0,00)). Ces précisions n'étant pas calculées sur l'ensemble des occurrences, pour pouvoir les comparer, la colonne $PG\%$ précise en pourcentage la *performance* qui combine le *gain* et le *rappel*. La colonne $PT\%$ indique en pourcentage la précision obtenue sur l'ensemble des occurrences (*i.e.* précision des classifieurs TPCM(0,00) et TNB(0,00)). Toutes ces mesures sont données pour la taille de contexte, précisée dans la colonne T, qui donne la meilleure précision $PT\%$.

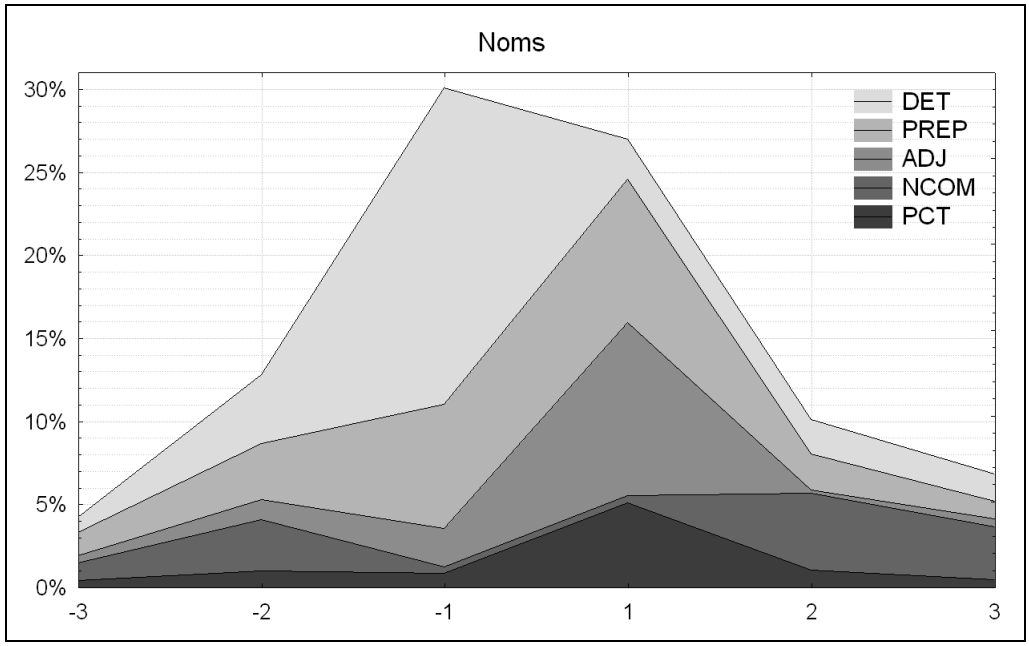


Figure 7.12 – Répartition spatiale, autour du mot à désambiguïser, des principales catégories grammaticales des indices utilisés pour lever l'ambiguïté des noms. La désambiguïsation a été effectuée par le classifieur TPCM(0,00) avec le critère *[jeton]-[ordonne]-[mot]* en considérant un contexte de plus ou moins trois mots.

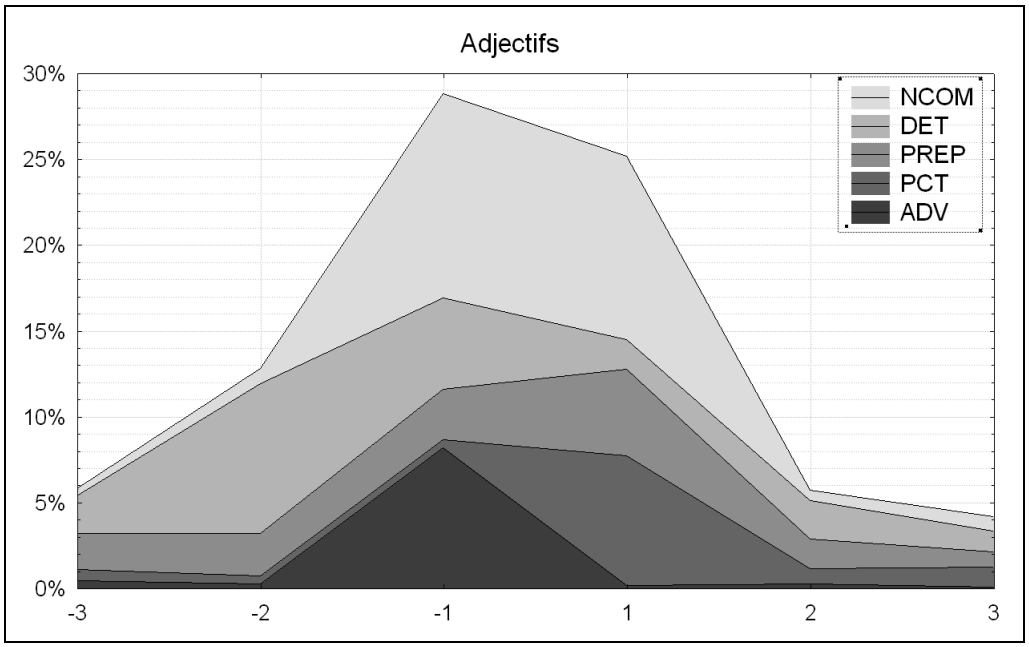


Figure 7.13 – Répartition spatiale, autour du mot à désambiguïser, des principales catégories grammaticales des indices utilisés pour lever l'ambiguïté des adjectifs. Ce graphique se lit de la même manière que celui de la figure 7.12.

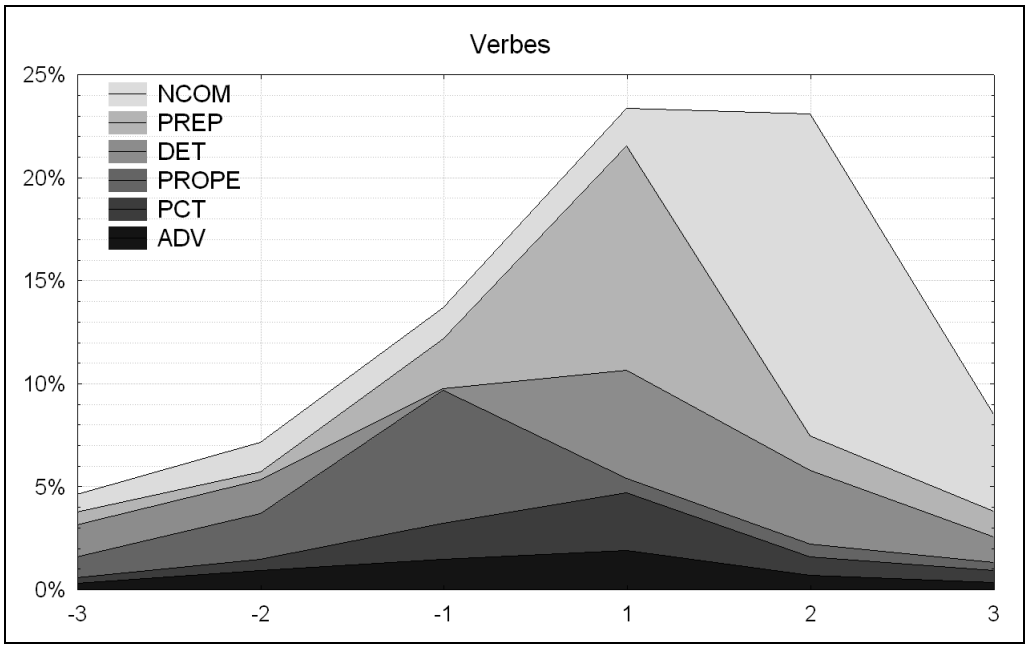


Figure 7.14 – Répartition spatiale, autour du mot à désambiguïser, des principales catégories grammaticales des indices utilisés pour lever l'ambiguïté des verbes. Ce graphique se lit de la même manière que celui de la figure 7.12.

verbes, les positions −2 et +2 en contiennent 30%. Nous comprenons très bien pourquoi la désambiguïsation des verbes nécessite, bien plus que les deux autres catégories grammaticales, un contexte d'au moins plus ou moins deux mots.

Le tableau 7.14 résume tout ce que nous venons de dire concernant l'information la plus utile pour la désambiguïsation de nos trois catégories grammaticales de vocables.

La dissymétrie de la figure des verbes, et plus précisément la forte prédominance des indices sélectionnés en position +1, +2 et +3 par rapport à leur position symétrique respective, incite à penser que la désambiguïsation des verbes se fait plus en fonction de leur objet que de leur sujet puisque la forme *sujet-verbe-complément* est en principe la forme majoritaire. Cette dissymétrie nous amène naturellement à nous poser la question de la pertinence de la symétrie de la taille du contexte. Le tableau 7.15 donne la précision obtenue par plusieurs contextes dissymétriques. Le contexte qui semble fonctionner le mieux en moyenne pour les deux classifieurs est le contexte *[-2 ; +4]*, montrant ainsi le bien-fondé de l'utilisation d'un contexte dissymétrique pour la désambiguïsation des verbes. L'amélioration ainsi obtenue est de 0,35% pour le classifieur TPCM(0,00) et de 0,53% pour le classifieur TNB(0,00).

Dans cette section, nous rejoignons sur plusieurs points les résultats de Yarowsky (1993), bien que son étude ne porte que sur des *pseudo-mots* (cf. section 2.5.3) ne possédant que deux « sens » :

- « Adjectives derive almost all of their disambiguating information from the nouns their modify (.98) »
- « Nouns are best disambiguated by directly adjacent adjectives or nouns, with the content word to the left indicating a single sense with 99% precision.»
- « Verbs, for example, derive more disambiguating information from their objects (.95) than from their subjects (.90). »

7.3.8 Considérations sur la fréquence des vocables et des lexies

Jusqu'à présent nous avons cherché à optimiser la précision moyenne. Mais il ne faut pas oublier que cette moyenne est une moyenne arithmétique pondérée (cf. section 5.2.5). Ainsi, la précision de la désambiguïsation d'un vocable comme *mettre*, avec ses 5246 occurrences éclipse presque complètement la précision obtenue pour un vocable comme *barrage* qui ne possède que 92 occurrences. Il est donc indispensable de regarder également la performance vocable par vocable plutôt que la performance moyenne. Pour les mêmes raisons, nous nous intéressons également à la précision obtenue pour chacune des lexies en fonction de leur fréquence.

Performance par vocables

Le critère qui fonctionne le mieux est *[lemme]-[ordonne]-[mot]*. Nous voulons savoir si ce critère fonctionne aussi bien pour les vocables qui sont peu fréquents que pour les vocables très fréquents.

La figure 7.15 montre la précision obtenue par les classifieurs TPCM(0,00) et TNB(0,00) en fonction de la fréquence des occurrences des vocables. Ce graphique montre clairement que la précision obtenue n'est pas fonction de la fréquence des vocables et est aussi bonne pour les vocables peu fréquents que pour les vocables plus fréquents. Nous observons également que cette constance, matérialisée par la courbe de régression logarithmique, n'est qu'un ajustement et que les fluctuations autour de cet

		Information utile		
		Noms	Adjectifs	Adverbes
Cible	Nom	$-2, +2$	$+1$	
	Adjectif	$-1, +1$		-1
	Verbes	$+2, +3$		

Tableau 7.14 – Information la plus utile pour la désambiguïsation de nos trois catégories grammaticales de vocables. La cible désigne l'occurrence à désambiguïser. Ce tableau donne, pour les occurrences de noms, d'adjectifs et de verbes, la catégorie grammaticale et la position par rapport à la cible des informations les plus utiles pour la levée de l'ambiguïté.

Contexte	TPCM(0,00)	TNB(0,00)
[-3 ; +3]	69,08%	71,84%
[-3 ; +4]	69,22%	71,95%
[-3 ; +5]	69,15%	71,48%
[-2 ; +3]	69,30%	72,39%
[-2 ; +4]	69,44%	72,37%
[-2 ; +5]	69,33%	71,85%

Tableau 7.15 – Précisions des classifieurs TPCM(0,00) et TNB(0,00) en utilisant le critère *[lemme]-[ordonne]-[mot]* avec un contexte dissymétrique pour les verbes. Dans la première colonne, *[-a ; +b]* identifie un contexte commençant au mot en position *-a* et se terminant en position *+b* par rapport au verbe à désambiguïser. La première ligne rappelle les meilleures précisions obtenues avec un contexte symétrique.

ajustement sont importantes. Nous sommes en tout cas rassuré sur les performances obtenues sur les vocables peu fréquents.

Le graphique de la figure 7.16 montre néanmoins que, si la précision ne semble pas fonction de la fréquence des vocables, le gain semble croître avec la fréquence des vocables. Cette différence de comportement entre la précision et le gain provient du fait que la précision de l'algorithme majoritaire a tendance à décroître quand la fréquence des vocables croît. Autrement dit, les vocables les plus fréquents ont une entropie de la distribution des fréquences des occurrences sur les lexies plus importante. L'explication de ce phénomène est fort simple : comme nous l'avons vu dans la section 4.5.4, l'entropie de la répartition des occurrences sur les lexies est en moyenne la plus importante pour les 20 verbes qui sont également les vocables les plus fréquents.

Sur les deux graphiques que nous venons de présenter nous observons que les deux classifieurs ont des comportements très semblables avec tout de même un avantage pour le classifieur TNB(0,00).

Performance par lexies

Nous nous intéressons maintenant à la précision des classifieurs TPCM(0,00) et TNB(0,00) en fonction de la fréquence des lexies.

Les figures 7.17 et 7.18 montrent la précision moyenne, en pourcentage, d'étiquetage correct par fréquence de lexie en distinguant les lexies majoritaires des lexies minoritaires. Ces graphiques permettent d'observer que :

- les précisions obtenues sur les lexies majoritaires sont nettement supérieures aux précisions obtenues sur les lexies minoritaires ;
- la précision va en croissant avec la fréquence des lexies de manière prononcée pour les lexies minoritaires, et de manière plus modérée pour les lexies majoritaires ;
- même les lexies dont la fréquence est très faible, de l'ordre de deux à cinq occurrences, obtiennent un pourcentage moyen d'étiquetage correct non négligeable (supérieur à 10% pour des lexies comportant deux occurrences et supérieur à 30% pour des lexies en comportant cinq) ;
- le classifieur TPCM(0,00) privilégie les classes majoritaires plus que le classifieur TNB(0,00) et obtient de meilleurs résultats pour ces classes que le classifieur TNB(0,00) ;
- la dispersion des précisions réalisées sur les différentes lexies est plus grande pour le classifieur TPCM(0,00) que sur le classifieur TNB(0,00) dont les résultats sont plus homogènes.

7.4 Critères basés sur les n-grammes ($n > 1$) évalués indépendamment

7.4.1 Introduction

Comme nous l'avons dit dans la définition 7.2, un n-gramme est une portion de texte constituée de n mots consécutifs. Les n-grammes pour $n = 1$ sont qualifiés d'unigramme, de bigramme pour $n = 2$ et de trigramme pour $n = 3$. En fait, les critères basés sur les unigrammes ont fait l'objet de la section précédente (7.3 Critères basés sur les cooccurrences évalués indépendamment) puisqu'une cooccurrence n'est, ni plus ni moins, qu'un unigramme. Les n-grammes, avec $n > 1$, sont des critères largement

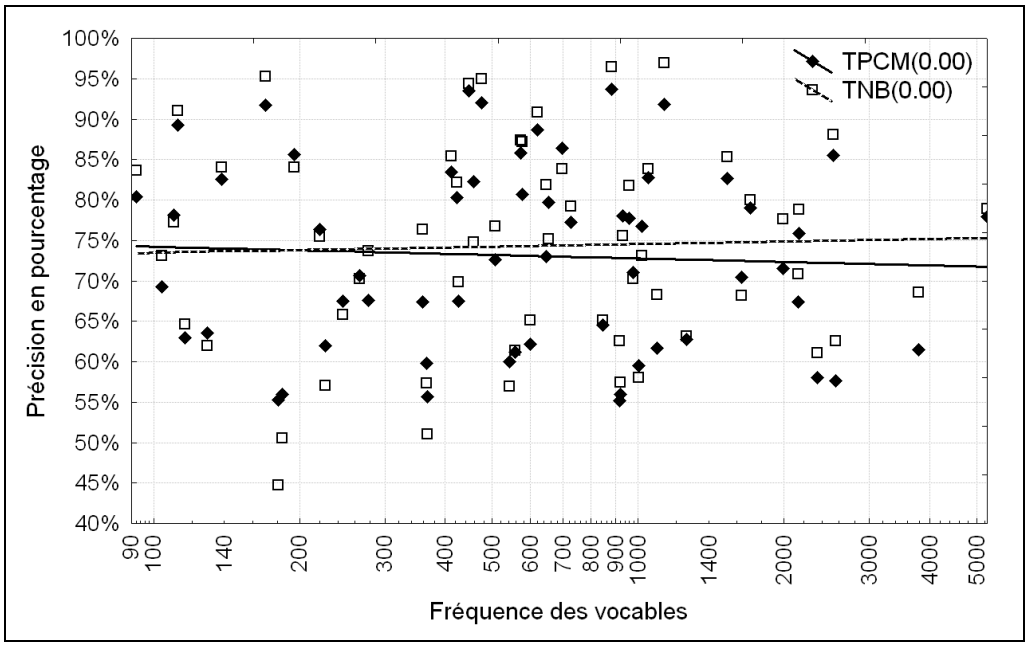


Figure 7.15 – Précision, obtenue par les classifieurs TPCM(0,00) et TNB(0,00), avec le critère *[lemme]-[ordonne]-[mot]* et un contexte de plus ou moins trois mots, en fonction de la fréquence des vocables. Pour faciliter la lecture, nous utilisons une échelle logarithmique en abscisse et avons ajouté à chacune des courbes de précision une courbe de régression logarithmique.

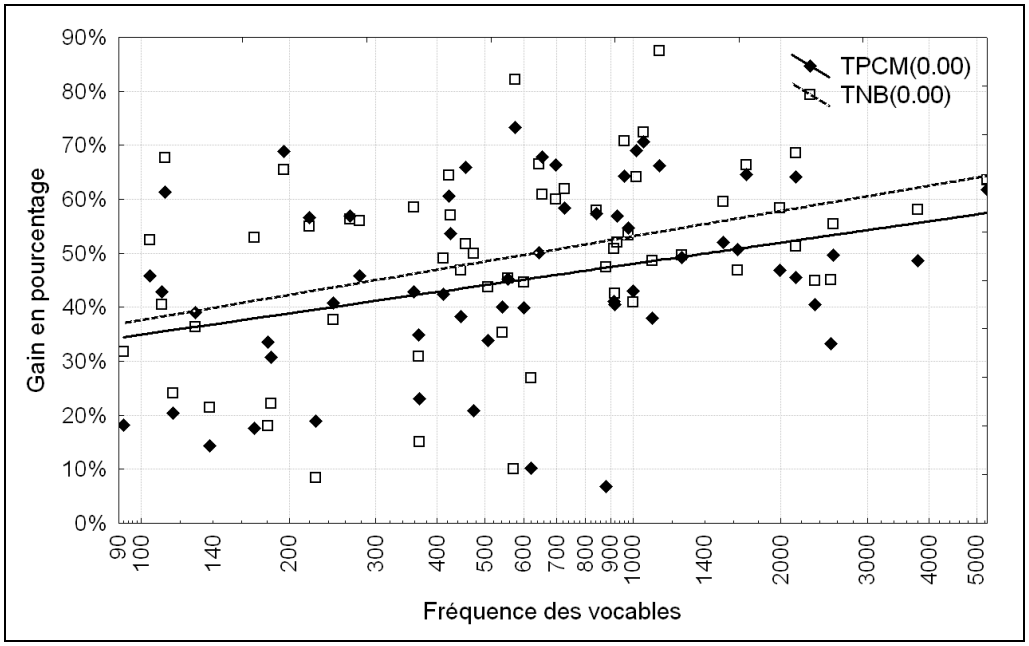


Figure 7.16 – Graphique identique au graphique 7.15 la précision étant remplacée par le gain.

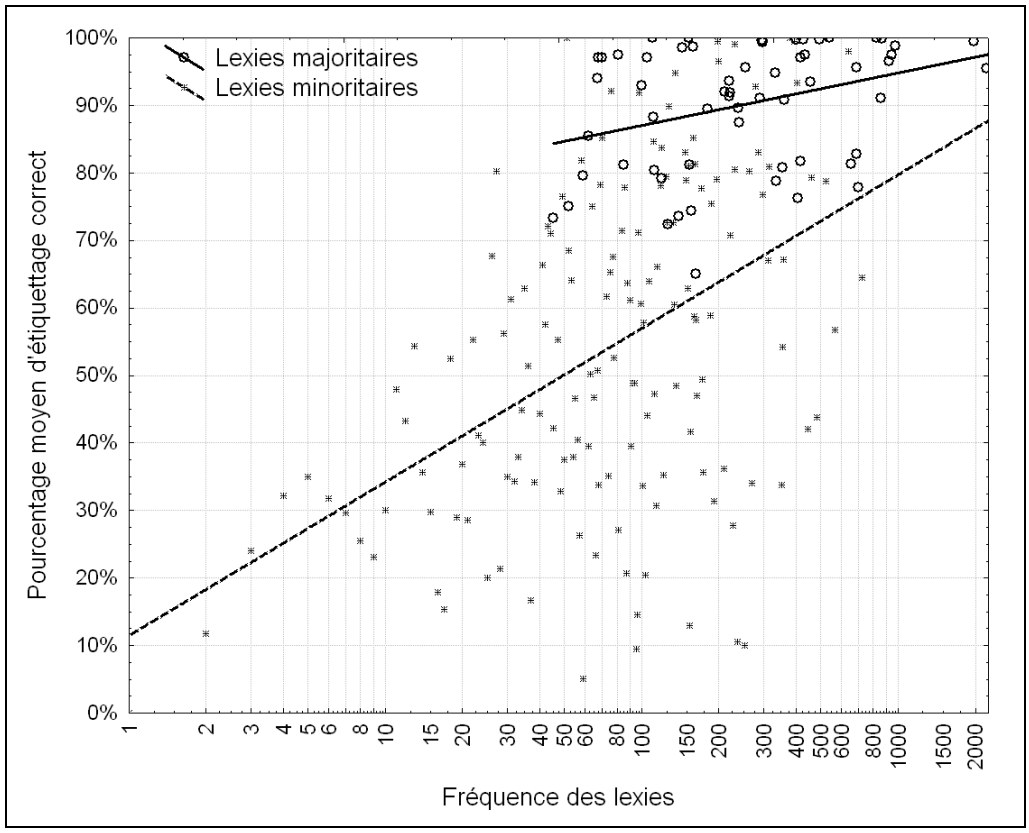


Figure 7.17 – Pourcentage moyen d'étiquetage correct, par fréquence de lexie, obtenu par le classifieur TPCM(0,00), avec le critère *[lemme]-[ordonne]-[mot]* et un contexte de plus ou moins trois mots. Pour faciliter la lecture, nous utilisons une échelle logarithmique en abscisse et avons ajouté deux courbes de régression logarithmique correspondant au nuage des points des lexies majoritaires et au nuage des points des lexies minoritaires.

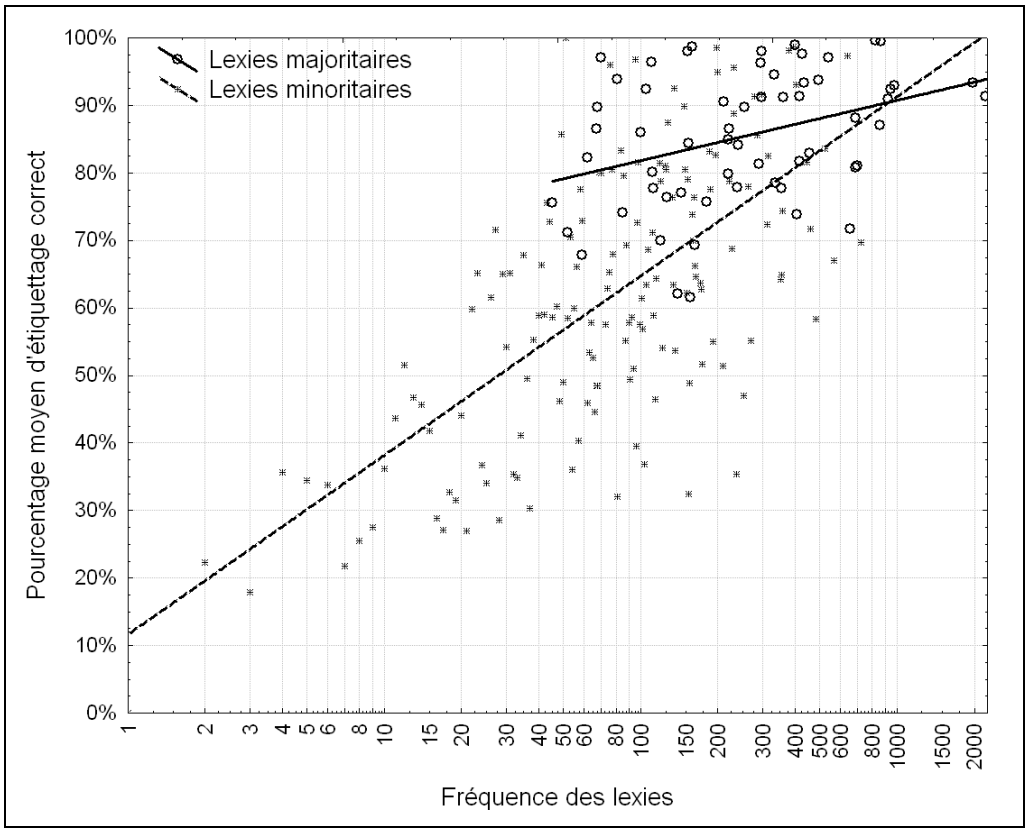


Figure 7.18 – Pourcentage moyen d'étiquetage correct, par fréquence de lexie, obtenu par le classifieur TNB(0,00), avec le critère *[lemme]-[ordonne]-[mot]* et un contexte de plus ou moins trois mots. Pour faciliter la lecture, nous utilisons une échelle logarithmique en abscisse et avons ajouté deux courbes de régression logarithmique correspondant au nuage des points des lexies majoritaires et au nuage des points des lexies minoritaires.

utilisés dans de nombreuses études (Yarowsky, 2000 ; Yarowsky *et al.*, 2001 ; Ng & Lee, 1996 ; Ng, 1997 ; Ng & Lee, 2002 ; Pedersen, 2001 ; Escudero *et al.*, 2000c ; etc.).

Dans cette étude, nous n'étudions pas les n-grammes pour toutes les valeurs de n. Nous nous concentrons sur les bigrammes et les trigrammes et faisons, lorsque cela nous paraît nécessaire, quelques incursions avec des valeurs de n supérieures à trois. Tout comme pour les cooccurrences, un critère basé sur les n-grammes peut prendre de multiples formes. Toutes ces formes ont été décrites dans la section 7.3.1 d'introduction aux critères basés sur les cooccurrences évalués indépendamment. Les n-grammes (avec n supérieur à deux) permettent de capturer des expressions ou des constructions plus ou moins figées. Aussi des questions viennent s'ajouter à celles que nous nous posions sur les cooccurrences (ou unigrammes) :

1. Au niveau des limites du contexte, devons-nous être plus restrictif que pour les cooccurrences et considérer qu'un n-gramme ne doit pas contenir de ponctuation, même faible ?
2. Lorsque nous faisons varier la taille du contexte, devons-nous aller jusqu'à ce que le mot à désambiguïser ne fasse plus partie du n-gramme ?

La question 1 paraît encore plus légitime pour les n-grammes, avec n supérieur à un, que pour les unigrammes. Nous nous intéressons à ce problème dans la section qui suit (section 7.4.2).

La question 2 est plus complexe. Certaines études (Ng, 1997 ; Escudero *et al.*, 2000c, 2000b ; etc.) s'imposent cette contrainte. Nous faisons remarquer à ce niveau que, l'étiquette *lemme* étant unique pour un vocable donné, pour les n-grammes composés de cette étiquette *lemme*, un n-gramme qui est adjacent au mot à désambiguïser véhicule exactement la même information que ce même n-gramme auquel est ajouté le mot à désambiguïser pour composer un (n+1)-gramme. Dans ce cas de figure, un n-gramme qui jouxte le mot à désambiguïser peut être assimilé à un (n+1)-gramme qui le contient. La question devient donc : les n-grammes doivent-ils contenir le mot à désambiguïser ou au moins lui être adjacents ? Nous tentons de répondre à cette question dans la section 7.4.5.

7.4.2 Quelles doivent-être les limites du contexte ?

Pour répondre à cette question, nous allons évaluer et comparer les performances de critères basés sur les bigrammes et les trigrammes dont le contexte est restreint de trois manières différentes :

1. un contexte non restreint qui accepte donc les mots en dehors de la phrase ;
2. un contexte restreint aux mots de la phrase, ponctuations comprises, excluant tous les mots en dehors de la phrase ;
3. un contexte restreint aux portions de texte ne contenant pas de ponctuation qui exclut donc toutes les ponctuations et tous les mots situés plus loin, par rapport au mot à désambiguïser, qu'une ponctuation faible ou forte.

Dans la section 7.3.4, nous avons observé que le critère qui fonctionne le mieux est *[lemme]-[ordonne]-[mot]* pour les unigrammes. Nous utilisons donc ce type de critère pour évaluer ceux basés sur les bigrammes et les trigrammes afin de comparer les trois limites de contextes. Nous notons ces critères *[2gr]-[lemme]-[ordonne]-[mot]* pour les bigrammes et *[3gr]-[lemme]-[ordonne]-[mot]* pour les trigrammes. Dans cette section, nous complétons ces deux dénominations par un astérisque (*) lorsque le contexte du critère est restreint de la première manière, deux astérisques lorsqu'il est restreint de la

seconde et trois lorsqu'il est restreint de la troisième. Nous obtenons ainsi six familles de critères :

1. *[2gr]-[lemme]-[ordonne]-[mot]** ;
2. *[2gr]-[lemme]-[ordonne]-[mot]*** ;
3. *[2gr]-[lemme]-[ordonne]-[mot]**** ;
4. *[3gr]-[lemme]-[ordonne]-[mot]** ;
5. *[3gr]-[lemme]-[ordonne]-[mot]*** ;
6. *[3gr]-[lemme]-[ordonne]-[mot]****.

Dans la section 7.3.4, nous avons observé que les tailles de contexte optimales sont très petites et toujours inférieures à plus ou moins quatre mots. Nous choisissons d'évaluer les six familles de critères que nous venons d'exposer pour des contextes allant de plus ou moins un mot à plus ou moins cinq mots.

Pour illustrer notre propos sur un exemple concret, prenons la portion de corpus suivante :

> « ... une saisie en dehors de la flagrance 33 , prolongation de la garde à *vue , placement en* **détention** *provisoire . En même* temps que l ' institution du juge d ' instruction ... »

Pour un contexte de plus ou moins trois mots et en travaillant sur le vocable *détention*, ces six critères retournent respectivement les indices suivants :

1. `[-4_vue_,]`, `[-3_,_placement]`, `[-2_placement_en]`, `[-1_en_détention]`, `[0_détention_provisoire]`, `[1_provisoire_.]`, `[2_._en]`, `[3_en_même]` ;
2. `[-4_vue_,]`, `[-3_,_placement]`, `[-2_placement_en]`, `[-1_en_détention]`, `[0_détention_provisoire]`, `[1_provisoire_.]` ;
3. `[-2_placement_en]`, `[-1_en_détention]`, `[0_détention_provisoire]` ;
4. `[-4_vue_,_placement]`, `[-3_,_placement_en]`, `[-2_placement_en_détention]`, `[-1_en_détention_provisoire]`, `[0_détention_provisoire_.]`, `[1_provisoire_._en]`, `[2_._en_même]` ;
5. `[-4_vue_,_placement]`, `[-3_,_placement_en]`, `[-2_placement_en_détention]`, `[-1_en_détention_provisoire]`, `[0_détention_provisoire_.]` ;
6. `[-2_placement_en_détention]`, `[-1_en_détention_provisoire]` ;

Cet exemple montre que nous avons choisi d'inclure le mot à désambiguïser dans les n-grammes à cheval sur ce mot de manière à ce que tous les n-grammes soient bien composés de n mots. Comme nous l'avons déjà évoqué dans l'introduction, dans certaines études (Ng, 1997 ; Escudero *et al.*, 2000c, 2000b ; etc.), tous les n-grammes considérés étant à cheval sur le mot à désambiguïser, ce mot est retiré des n-grammes. Dans la section 7.4.5 nous abordons le problème de savoir s'il est plus pertinent de ne considérer que des n-grammes à cheval sur le mot à désambiguïser ou pas.

Le tableau 7.16 synthétise les résultats de cette étude. Tout comme pour les unigrammes, il semble que, pour les bigrammes et les trigrammes, le plus judicieux soit de se restreindre au contexte de la phrase. Nous remarquons toutefois que la différence entre se restreindre au contexte de la phrase (critères **) et accepter les mots en dehors de la phrase (critères *), est très faible. En revanche, se restreindre aux portions de texte ne contenant pas de ponctuation (critères ***) peut entraîner une dégradation

Classifieur	Famille de critères	Précision	Contexte
TPCM(0,00)	*[2gr]-[lemme]-[ordonne]-[mot]**	73,98%	3
	*[2gr]-[lemme]-[ordonne]-[mot]***	74,07%	3
	*[2gr]-[lemme]-[ordonne]-[mot]****	73,89%	4
TNB(0,00)	*[2gr]-[lemme]-[ordonne]-[mot]**	75,50%	3
	*[2gr]-[lemme]-[ordonne]-[mot]***	75,68%	3
	*[2gr]-[lemme]-[ordonne]-[mot]****	75,06%	4
TPCM(0,00)	*[3gr]-[lemme]-[ordonne]-[mot]**	72,88%	4
	*[3gr]-[lemme]-[ordonne]-[mot]***	72,86%	4
	*[3gr]-[lemme]-[ordonne]-[mot]****	71,65%	4
TNB(0,00)	*[3gr]-[lemme]-[ordonne]-[mot]**	73,05%	4
	*[3gr]-[lemme]-[ordonne]-[mot]***	73,10%	4
	*[3gr]-[lemme]-[ordonne]-[mot]****	71,82%	4

Tableau 7.16 – Impact de différentes limitations de contexte sur des critères basés sur les n-grammes. Dans la seconde colonne, * signifie que le contexte n'est pas restreint, ** signifie que le contexte est restreint aux mots de la phrase et *** qu'il est restreint aux portions de texte ne contenant pas de ponctuations (forte ou faible). La colonne *Précision* indique la précision atteinte pour la taille de contexte, indiquée dans la colonne *Contexte*, qui donne la meilleure précision.

relativement importante de la précision. Les ponctuations faibles ne devraient donc pas constituer une limite pour des n-grammes avec $n > 1$.

7.4.3 Définitions et évaluation de 48 critères

Définitions des 48 critères

Dans la section 7.3.4 nous avons défini 24 familles de critères d'unigrammes. Nous définissons ici les mêmes familles de critères pour les bigrammes et les trigrammes et nous les désignons de la manière suivante : *[<param0>]-[<param1>]-[<param2>]-[<param3>]*. Tout comme dans la section sur les unigrammes, *<param3>* peut prendre deux valeurs, *mot* et *mot-plein* suivant que le critère considère tous les mots ou seulement les mots pleins. *<param2>* peut prendre trois valeurs, *ordonne*, *differencie* et *non-ordonne* selon que le critère tienne compte de l'ordre des n-grammes (*ordonne*), qu'il différencie simplement le contexte droit du gauche (*differencie*) ou qu'il ne tienne pas compte de la position de la tête des n-grammes (*non-ordonne*) par rapport au mot à désambiguïser. *<param1>* peut prendre quatre valeurs, *jeton*, *lemme*, *ems* et *smallems* suivant que le critère utilise la forme brute des mots (*jeton*), leur lemme (*lemme*), leur étiquette morphosyntaxique (*ems*) ou leur étiquette morphosyntaxique simplifiée (*smallems*). Enfin, *<param0>* peut prendre deux valeurs, *2gr* si les n-grammes sont des bigrammes et *3gr* s'il s'agit de trigrammes.

Contrairement aux unigrammes, nous avons choisi de ne retenir aucun cas particulier hormis la restriction du contexte à celui de la phrase. Dans le cas des unigrammes, nous prenons en compte le mot à désambiguïser uniquement dans le cas où *<param1>=ems*, pour les bigrammes et les trigrammes le mot à désambiguïser est traité comme les autres de manière à ce que les bigrammes soient toujours constitués de deux mots et les trigrammes de trois. Dans le cas où *<param1>=jeton*, nous juxtaposons à l'étiquette *jeton* l'étiquette *ems* pour lever toute ambiguïté sur l'étiquette *jeton*. Nous n'effectuons plus cette juxtaposition pour les bigrammes ou les trigrammes puisque, dans la plupart

Nom des critères (sans le préfixe [2gr]-)		Noms		Adjectifs		Verbes		**Moyennes**	
Complet	Abrev.	$P\%$	T	$P\%$	T	$P\%$	T	$P\%$	T
[jeton]- [ordonne] -[mot]	JOM	82,0	4	74,9	2	70,3	4	73,5	4
[lemme]- [ordonne] -[mot]	LOM	81,5	4	76,4	3	71,0	3	74,1	3
[ems]- [ordonne] -[mot]	EOM	72,6	2	63,6	2	61,3	2	64,0	2
[smallems]- [ordonne] -[mot]	SOM	66,1	2	59,8	2	55,2	3	58,1	3
[jeton]- [ordonne] -[mot-plein]	JOMp	78,9	3	71,5	2	56,8	3	63,9	3
[lemme]- [ordonne] -[mot-plein]	LOMp	79,7	3	72,6	3	60,8	2	66,7	2
[ems]- [ordonne] -[mot-plein]	EOMp	67,5	1	58,6	1	52,3	2	56,5	1
[smallems]- [ordonne] -[mot-plein]	SOMp	61,4	2	54,3	2	47,6	2	51,6	2
[jeton]-[non-ordonne]-[mot]	JNoM	81,4	3	74,6	2	70,9	4	73,7	4
[lemme]-[non-ordonne]-[mot]	LNoM	80,7	4	75,9	2	71,1	4	73,8	3
[ems]-[non-ordonne]-[mot]	ENoM	70,8	2	62,8	2	59,1	2	62,1	2
[smallems]-[non-ordonne]-[mot]	SNoM	63,9	1	58,8	1	52,7	2	55,9	2
[jeton]-[non-ordonne]-[mot-plein]	JNoMp	79,1	3	71,5	2	57,7	3	64,4	3
[lemme]-[non-ordonne]-[mot-plein]	LNoMp	79,7	2	72,6	3	61,4	3	67,1	3
[ems]-[non-ordonne]-[mot-plein]	ENoMp	66,7	1	58,4	1	51,5	1	55,8	1
[smallems]-[non-ordonne]-[mot-plein]	SNoMp	60,9	1	53,3	1	46,0	1	50,3	1
[jeton]- [differencie] -[mot]	JDM	82,0	4	75,0	3	71,3	4	74,1	4
[lemme]- [differencie] -[mot]	LDM	81,5	4	76,3	3	72,0	4	74,6	4
[ems]- [differencie] -[mot]	EDM	72,5	2	63,6	2	61,2	2	64,0	2
[smallems]- [differencie] -[mot]	SDM	66,0	2	59,5	2	54,9	2	58,0	2
[jeton]- [differencie] -[mot-plein]	JDMp	79,1	3	71,5	2	57,7	5	64,4	5
[lemme]- [differencie] -[mot-plein]	LDMp	79,8	3	72,7	3	61,4	4	67,0	3
[ems]- [differencie] -[mot-plein]	EDMp	66,7	1	58,4	1	51,8	2	55,8	2
[smallems]- [differencie] -[mot-plein]	SDMp	61,2	2	53,7	2	46,7	2	50,9	2
Performance optimale		82,0%	4	76,4%	3	72,0%	4	74,6%	4
Précision de l'algorithme majoritaire		57,3%		46,4%		37,2%		42,9%	

Tableau 7.17 – Performance du classifieur TPCM(0,00) pour les 24 critères basés sur les bigrammes et les trois catégories grammaticales. La colonne *Abrev.* contient le nom abrégé du critère. Dans la colonne $P\%$ est mentionnée la précision maximale atteinte par le critère pour une taille de contexte précisée dans la colonne T.

des cas, les mots contigus incorporés au n-gramme permettent de lever l'ambiguïté sur l'étiquette *jeton*. En utilisant le critère *[2gr]-[jeton]-[ordonne]-[mot]* le gain obtenu par cette juxtaposition, par rapport à l'utilisation de l'étiquette *jeton* seule, est proche de zéro par valeur supérieure pour le classifieur TPCM(0,00), et proche de zéro par valeur inférieure pour le classifieur TNB(0,00).

Dans le cas où *<param2>=differencie*, nous distinguons chaque n-gramme suivant :

- qu'il se situe à gauche du mot à désambiguïser ;
- qu'il se situe à droite du mot à désambiguïser ;
- qu'il est à cheval sur le mot à désambiguïser.

Nous évaluons toutes ces familles de critères pour des contextes allant de plus ou moins un mot à plus ou moins huit mots.

Évaluation des performances

Les tableaux 7.17 et 7.18 indiquent les meilleures précisions atteintes par les classifieurs TPCM(0,00) et TNB(0,00) pour les 24 critères basés sur les bigrammes, et les trois catégories grammaticales.

Nom des critères (sans le préfixe [2gr]-)		Noms		Adjectifs		Verbes		**Moyennes**	
Complet	Abrev.	$P\%$	T	$P\%$	T	$P\%$	T	$P\%$	T
[jeton]- [ordonne] -[mot]	JOM	82,8	4	76,2	2	71,4	4	74,5	4
[lemme]- [ordonne] -[mot]	LOM	83,1	3	77,3	3	72,8	3	75,7	3
[ems]- [ordonne] -[mot]	EOM	74,0	4	64,1	3	63,5	4	65,8	3
[smallems]- [ordonne] -[mot]	SOM	67,9	4	60,1	1	58,8	5	60,8	5
[jeton]- [ordonne] -[mot-plein]	JOMp	79,3	2	71,5	2	56,7	2	63,9	2
[lemme]- [ordonne] -[mot-plein]	LOMp	79,4	2	72,6	3	60,4	2	66,4	2
[ems]- [ordonne] -[mot-plein]	EOMp	68,1	1	58,1	1	51,3	2	56,0	1
[smallems]- [ordonne] -[mot-plein]	SOMp	61,5	1	54,6	1	47,2	1	51,5	1
[jeton]-[non-ordonne]-[mot]	JNoM	82,9	3	76,2	2	72,3	4	75,0	4
[lemme]-[non-ordonne]-[mot]	LNoM	83,2	3	77,7	3	73,5	4	76,1	3
[ems]-[non-ordonne]-[mot]	ENoM	72,1	1	63,0	3	61,7	2	64,1	2
[smallems]-[non-ordonne]-[mot]	SNoM	66,8	2	60,0	1	57,7	3	59,5	3
[jeton]-[non-ordonne]-[mot-plein]	JNoMp	79,0	2	71,6	2	57,2	3	64,0	3
[lemme]-[non-ordonne]-[mot-plein]	LNoMp	79,3	2	72,5	2	60,5	2	66,4	2
[ems]-[non-ordonne]-[mot-plein]	ENoMp	67,3	1	57,9	1	50,7	1	55,4	1
[smallems]-[non-ordonne]-[mot-plein]	SNoMp	60,7	1	54,0	1	46,7	1	50,9	1
[jeton]- [differencie] -[mot]	JDM	83,2	4	76,3	3	72,7	4	75,4	4
[lemme]- [differencie] -[mot]	LDM	83,6	4	77,9	3	74,0	4	76,5	4
[ems]- [differencie] -[mot]	EDM	74,0	3	64,3	3	63,5	3	65,9	3
[smallems]- [differencie] -[mot]	SDM	68,9	4	60,0	1	59,3	4	61,5	4
[jeton]- [differencie] -[mot-plein]	JDMp	79,3	2	71,5	2	57,1	3	64,1	3
[lemme]- [differencie] -[mot-plein]	LDMp	79,4	2	72,6	2	60,6	3	66,4	3
[ems]- [differencie] -[mot-plein]	EDMp	67,3	1	57,9	1	51,3	3	55,4	1
[smallems]- [differencie] -[mot-plein]	SDMp	60,7	1	54,0	1	47,4	3	50,9	1
Performance optimale		83,6%	4	77,9%	3	74,0%	4	76,5%	4
Précision de l'algorithme majoritaire		57,3%		46,4%		37,2%		42,9%	

Tableau 7.18 – Performance du classifieur TNB(0,00) pour les 24 critères basés sur les bigrammes. Ce tableau se lit de la même manière que le tableau 7.17.

TPCM(0,00)				TNB(0,00)			
Noms	Adjectifs	Verbes	**Moyennes**	Noms	Adjectifs	Verbes	**Moyennes**
JOM	LOM	LDM	LDM	LDM	LDM	LDM	LDM
JDM	LDM	JDM	JDM	JDM	LNoM	LNoM	LNoM
LOM	LNoM	LNoM	LOM	LNoM	LOM	LOM	LOM
LDM	JDM	LOM	LNoM	LOM	JDM	JDM	JDM
JNoM	JOM	JNoM	JNoM	JNoM	JNoM	JNoM	JNoM
LNoM	JNoM	JOM	JOM	JOM	JOM	JOM	JOM
LDMp	LDMp	LNoMp	LNoMp	LDMp	LOMp	EOM	LDMp
LNoMp	LOMp	LDMp	LDMp	LOMp	LDMp	EDM	LNoMp
LOMp	LNoMp	EOM	LOMp	JDMp	LNoMp	ENoM	LOMp
JNoMp	JNoMp	EDM	JNoMp	JOMp	JNoMp	LDMp	EDM
JDMp	JOMp	LOMp	JDMp	LNoMp	JOMp	LNoMp	EOM
JOMp	JDMp	ENoM	EOM	JNoMp	JDMp	LOMp	ENoM
EOM	EOM	JDMp	EDM	EOM	EDM	SDM	JDMp
EDM	EDM	JNoMp	JOMp	EDM	EOM	SOM	JNoMp
ENoM	ENoM	JOMp	ENoM	ENoM	ENoM	SNoM	JOMp
EOMp	SOM	SOM	SOM	SDM	SOM	JNoMp	SDM
ENoMp	SDM	SDM	SDM	EOMp	SNoM	JDMp	SOM
EDMp	SNoM	SNoM	EOMp	SOM	SDM	JOMp	SNoM
SOM	EOMp	EOMp	SNoM	ENoMp	EOMp	EOMp	EOMp
SDM	ENoMp	EDMp	EDMp	EDMp	ENoMp	EDMp	ENoMp
SNoM	EDMp	ENoMp	ENoMp	SNoM	EDMp	ENoMp	EDMp
SOMp	SOMp	SOMp	SOMp	SOMp	SOMp	SDMp	SOMp
SDMp	SDMp	SDMp	SDMp	SNoMp	SNoMp	SOMp	SNoMp
SNoMp	SNoMp	SNoMp	SNoMp	SDMp	SDMp	SNoMp	SDMp

Tableau 7.19 – Les 24 critères basés sur les bigrammes classés par ordre de précision décroissante pour les trois catégories grammaticales et pour les deux classifieurs TPCM(0,00) et TNB(0,00).

Nom des critères (sans le préfixe [3gr]-)		Noms		Adjectifs		Verbes		**Moyennes**	
Complet	Abrev.	*P*%	T	*P*%	T	*P*%	T	*P*%	T
[jeton]- [ordonne] -[mot]	JOM	80,9	4	69,7	3	64,2	6	68,6	4
[lemme]- [ordonne] -[mot]	LOM	81,3	4	72,7	3	70,1	4	72,9	4
[ems]- [ordonne] -[mot]	EOM	74,4	3	64,3	3	62,7	3	65,4	3
[smallems]- [ordonne] -[mot]	SOM	68,1	3	61,4	3	58,6	3	61,1	3
[jeton]- [ordonne] -[mot-plein]	JOMp	68,5	3	53,6	4	44,5	4	51,1	4
[lemme]- [ordonne] -[mot-plein]	LOMp	69,7	3	55,9	2	48,0	3	53,9	2
[ems]- [ordonne] -[mot-plein]	EOMp	66,4	2	56,4	2	48,9	3	53,8	2
[smallems]- [ordonne] -[mot-plein]	SOMp	61,0	2	54,2	2	48,1	2	51,8	2
[jeton]-[non-ordonne]-[mot]	JNoM	81,1	4	69,6	3	65,1	6	69,1	5
[lemme]-[non-ordonne]-[mot]	LNoM	81,4	4	72,5	3	70,6	5	73,1	5
[ems]-[non-ordonne]-[mot]	ENoM	73,5	2	63,5	2	61,1	3	63,9	3
[smallems]-[non-ordonne]-[mot]	SNoM	66,6	2	60,2	2	57,1	2	59,6	2
[jeton]-[non-ordonne]-[mot-plein]	JNoMp	68,8	4	54,0	6	45,0	6	51,5	4
[lemme]-[non-ordonne]-[mot-plein]	LNoMp	70,0	3	56,2	8	48,3	4	54,2	4
[ems]-[non-ordonne]-[mot-plein]	ENoMp	65,8	1	56,4	2	48,1	2	53,2	2
[smallems]-[non-ordonne]-[mot-plein]	SNoMp	60,4	1	54,0	2	47,8	1	51,4	1
[jeton]- [differencie] -[mot]	JDM	81,2	5	69,7	4	65,6	7	69,5	6
[lemme]- [differencie] -[mot]	LDM	81,7	8	72,6	5	71,1	6	73,5	6
[ems]- [differencie] -[mot]	EDM	74,3	3	64,3	3	62,5	3	65,3	3
[smallems]- [differencie] -[mot]	SDM	67,8	3	60,8	3	58,3	3	60,7	3
[jeton]- [differencie] -[mot-plein]	JDMp	68,6	4	53,8	8	44,9	8	51,4	8
[lemme]- [differencie] -[mot-plein]	LDMp	69,8	3	56,1	4	48,3	8	54,1	4
[ems]- [differencie] -[mot-plein]	EDMp	65,8	1	56,4	2	48,4	3	53,3	3
[smallems]- [differencie] -[mot-plein]	SDMp	60,4	1	54,0	2	47,8	1	51,4	1
Performance optimale		81,7%	8	72,7%	3	71,1%	6	73,5%	6
Précision de l'algorithme majoritaire		57,3%		46,4%		37,2%		42,9%	

Tableau 7.20 – Performance du classifieur TPCM(0,00) pour les 24 critères basés sur les trigrammes. Ce tableau se lit de la même manière que le tableau 7.17.

Nom des critères (sans le préfixe [3gr]-)		Noms		Adjectifs		Verbes		**Moyennes**	
Complet	Abrev.	$P\%$	T	$P\%$	T	$P\%$	T	$P\%$	T
[jeton]- [ordonne] -[mot]	JOM	81,3	4	69,7	3	64,5	4	68,9	4
[lemme]- [ordonne] -[mot]	LOM	81,9	4	72,7	3	70,2	4	73,1	4
[ems]- [ordonne] -[mot]	EOM	74,2	4	63,3	6	62,4	4	65,0	4
[smallems]- [ordonne] -[mot]	SOM	67,6	5	59,9	1	58,8	5	60,7	5
[jeton]- [ordonne] -[mot-plein]	JOMp	68,6	4	53,7	4	44,5	4	51,1	4
[lemme]- [ordonne] -[mot-plein]	LOMp	69,9	3	55,9	3	48,0	2	54,0	2
[ems]- [ordonne] -[mot-plein]	EOMp	66,8	1	55,4	2	47,8	3	52,6	2
[smallems]- [ordonne] -[mot-plein]	SOMp	61,0	1	54,2	1	48,1	1	51,8	1
[jeton]-[non-ordonne]-[mot]	JNoM	81,5	4	69,7	3	65,5	5	69,5	5
[lemme]-[non-ordonne]-[mot]	LNoM	82,2	4	72,7	3	71,1	5	73,7	5
[ems]-[non-ordonne]-[mot]	ENoM	73,2	2	62,9	3	61,2	4	63,9	3
[smallems]-[non-ordonne]-[mot]	SNoM	66,7	2	59,9	1	57,8	4	59,2	4
[jeton]-[non-ordonne]-[mot-plein]	JNoMp	68,8	4	54,2	8	44,9	4	51,5	4
[lemme]-[non-ordonne]-[mot-plein]	LNoMp	70,0	3	56,2	4	48,2	2	54,1	2
[ems]-[non-ordonne]-[mot-plein]	ENoMp	66,8	1	55,6	2	47,2	1	52,6	1
[smallems]-[non-ordonne]-[mot-plein]	SNoMp	61,0	1	54,2	1	48,1	1	51,8	1
[jeton]- [differencie] -[mot]	JDM	81,6	4	69,8	5	65,5	5	69,6	5
[lemme]- [differencie] -[mot]	LDM	82,3	5	72,7	3	71,2	5	73,8	5
[ems]- [differencie] -[mot]	EDM	74,3	4	63,5	7	62,6	4	65,1	4
[smallems]- [differencie] -[mot]	SDM	68,7	5	59,9	6	59,6	5	61,4	5
[jeton]- [differencie] -[mot-plein]	JDMp	68,8	4	53,9	4	44,8	6	51,4	4
[lemme]- [differencie] -[mot-plein]	LDMp	70,0	4	56,1	4	48,2	2	54,1	2
[ems]- [differencie] -[mot-plein]	EDMp	66,8	1	55,6	2	47,3	3	52,6	1
[smallems]- [differencie] -[mot-plein]	SDMp	61,0	1	54,2	1	48,1	1	51,8	1
Performance optimale		82,3%	5	72,7%	3	71,2%	5	73,8%	5
Précision de l'algorithme majoritaire		57,3%		46,4%		37,2%		42,9%	

Tableau 7.21 – Performance du classifieur TNB(0,00) pour les 24 critères basés sur les trigrammes. Ce tableau se lit de la même manière que le tableau 7.17.

TPCM(0,00)				TNB(0,00)			
Noms	Adjectifs	Verbes	**Moyennes**	Noms	Adjectifs	Verbes	**Moyennes**
LDM	LOM	LDM	LDM	LDM	LDM	LDM	LDM
LNoM	LDM	LNoM	LNoM	LNoM	LNoM	LNoM	LNoM
LOM	LNoM	LOM	LOM	LOM	LOM	LOM	LOM
JDM	JOM	JDM	JDM	JDM	JDM	JNoM	JDM
JNoM	JDM	JNoM	JNoM	JNoM	JOM	JDM	JNoM
JOM	JNoM	JOM	JOM	JOM	JNoM	JOM	JOM
EOM	EOM	EOM	EOM	EDM	EDM	EDM	EDM
EDM	EDM	EDM	EDM	EOM	EOM	EOM	EOM
ENoM	ENoM	ENoM	ENoM	ENoM	ENoM	ENoM	ENoM
LNoMp	SOM	SOM	SOM	LNoMp	SDM	SDM	SDM
LDMp	SDM	SDM	SDM	LDMp	SOM	SOM	SOM
LOMp	SNoM	SNoM	SNoM	LOMp	SNoM	SNoM	SNoM
JNoMp	EOMp	EOMp	LNoMp	JNoMp	LNoMp	LNoMp	LNoMp
JDMp	ENoMp	EDMp	LDMp	JDMp	LDMp	LDMp	LDMp
JOMp	EDMp	LDMp	LOMp	SDM	LOMp	SOMp	LOMp
SOM	LNoMp	LNoMp	EOMp	JOMp	ENoMp	SNoMp	EOMp
SDM	LDMp	ENoMp	EDMp	SOM	EDMp	SDMp	ENoMp
SNoM	LOMp	SOMp	ENoMp	EOMp	EOMp	LOMp	EDMp
EOMp	SOMp	LOMp	SOMp	ENoMp	SOMp	EOMp	SOMp
ENoMp	SNoMp	SNoMp	JNoMp	EDMp	SNoMp	EDMp	SNoMp
EDMp	SDMp	SDMp	JDMp	SNoM	SDMp	ENoMp	SDMp
SOMp	JNoMp	JNoMp	SNoMp	SOMp	JNoMp	JNoMp	JNoMp
SNoMp	JDMp	JDMp	SDMp	SNoMp	JDMp	JDMp	JDMp
SDMp	JOMp	JOMp	JOMp	SDMp	JOMp	JOMp	JOMp

Tableau 7.22 – Les 24 critères basés sur les trigrammes classés par ordre de précision décroissante pour les trois catégories grammaticales et pour les deux classifieurs TPCM(0,00) et TNB(0,00).

Le tableau 7.19 montre les 24 critères basés sur les bigrammes classés par ordre de précision décroissante pour les trois catégories grammaticales et pour les deux classifieurs TPCM(0,00) et TNB(0,00).

Les tableaux 7.20 et 7.21 indiquent les meilleures précisions atteintes par les classifieurs TPCM(0,00) et TNB(0,00) pour les 24 critères basés sur les trigrammes, et les trois catégories grammaticales.

Le tableaux 7.22 montre les 24 critères basés sur les trigrammes classés par ordre de précision décroissante pour les trois catégories grammaticales et pour les deux classifieurs TPCM(0,00) et TNB(0,00).

Ces trois tableaux pour les bigrammes (tableaux 7.17, 7.18 et 7.19) et trois tableaux pour les trigrammes (tableaux 7.20, 7.21 et 7.22) sont à rapprocher de leur trois analogues (tableaux 7.4, 7.5 et 7.6) pour les unigrammes. Nous détaillons et illustrons les performances de ces critères de manière plus ciblée dans les sections qui suivent. Cependant, nous pouvons déjà donner un certain nombre d'informations remarquables à la simple inspection de ces tableaux :

- L'information la plus notable est probablement la comparaison entre les meilleures précisions atteintes par des critères basés sur les unigrammes (72,3% et 74,5% pour les classifieurs TPCM(0,00) et TNB(0,00)), les meilleures précisions atteintes par des critères basés sur les bigrammes (74,6% et 76,5%) et les meilleures précisions atteintes par des critères basés sur les trigrammes (73,5% et 73,8%). Il est surprenant d'observer que la précision obtenue en n'utilisant que des trigrammes est comparable à celle obtenue avec des unigrammes. Mieux encore, la précision obtenue en utilisant seulement le meilleur critère basé sur les bigrammes est nettement supérieure à celle obtenue avec le meilleur critère basé sur les unigrammes. Nous discutons de ce phénomène dans la section 7.4.6.
- Nous observons que le contexte optimal est en moyenne plus grand pour les trigrammes (plus ou moins cinq à six mots) que pour les bigrammes (plus ou moins quatre mots) lui même plus grand que pour les unigrammes (plus ou moins trois mots). À ce sujet, la taille de contexte optimale pour des trigrammes et pour la catégorie des noms (tableau 7.20) est de plus ou moins huit mots. Pour ce critère, nous avons poussé le contexte jusqu'à plus ou moins 12 mots afin de confirmer ce résultat.
- Nous remarquons que si certains critères qui ne considèrent que les mots pleins fonctionnent bien pour les unigrammes, ils fonctionnent moins bien pour les bigrammes et encore nettement moins bien pour les trigrammes. Par exemple, le critère *[jeton]-[ordonne]-[mot-plein]* obtient une précision moyenne de 66,4% avec des unigrammes, 63,9% avec des bigrammes, et seulement 51,1% avec des trigrammes. Nous revenons sur ce phénomène dans la section 7.4.4.
- Enfin, en regardant la taille optimale de contexte des meilleurs critères, nous observons que cette taille est systématiquement suffisamment grande pour que les n-grammes les plus éloignés ne contiennent pas la cible. Nous revenons sur cette particularité dans la section 7.4.5.

7.4.4 Impacts des différents paramètres par rapport aux unigrammes

Le tableau 7.23 montre l'impact du choix de chacun des trois paramètres *[<param1>]*, *[<param2>]* et *[<param3>]* des 24 critères étudiés basés sur les unigrammes, puis sur les bigrammes et enfin sur les trigrammes en utilisant le classifieur TPCM(0,00).

Unigrammes	**Noms**	**Adjectifs**	**Verbes**	**Moy**
[jeton]-	16,3%	17,8%	18,8%	18,0%
[lemme]-	16,6%	18,2%	18,5%	17,9%
[ems]-	4,4%	2,9%	4,4%	4,1%
[smallems]-	0,0%	0,0%	0,0%	0,0%
-[ordonne]-	2,9%	1,1%	2,2%	2,3%
-[differencie]-	1,9%	0,8%	1,6%	1,3%
-[non-ordonne]-	0,0%	0,0%	0,0%	0,0%
-[mot]	0,3%	2,5%	6,9%	4,7%
-[mot-plein]	0,0%	0,0%	0,0%	0,0%
Bigrammes	**Noms**	**Adjectifs**	**Verbes**	**Moy**
[jeton]-	17,2%	16,6%	13,6%	14,9%
[lemme]-	17,2%	17,8%	15,8%	16,4%
[ems]-	6,2%	4,3%	5,7%	5,6%
[smallems]-	0%	0%	0%	0%
-[ordonne]-	0,6%	0,4%	−0,6%	−0,2%
-[differencie]-	0,5%	0,3%	0,4%	0,5%
-[non-ordonne]-	0%	0%	0%	0%
-[mot]	2,7%	3,4%	13,5%	9,5%
-[mot-plein]	0%	0%	0%	0%
Trigrammes	**Noms**	**Adjectifs**	**Verbes**	**Moy**
[jeton]-	10,8%	4,3%	1,9%	4,2%
[lemme]-	11,6%	6,9%	6,4%	7,6%
[ems]-	6%	2,7%	2,3%	3,1%
[smallems]-	0%	0%	0%	0%
-[ordonne]-	−0,2%	0,1%	−0,9%	−0,5%
-[differencie]-	0,2%	0,1%	0,5%	0,4%
-[non-ordonne]-	0%	0%	0%	0%
-[mot]	12,4%	15,9%	20,2%	17,7%
-[mot-plein]	0%	0%	0%	0%

Tableau 7.23 – Évaluation de l'impact du choix de chacun des trois paramètres */<param1>/*, */<param2>/* et */<param3>/* des 24 critères étudiés basés sur les unigrammes, puis sur les bigrammes et enfin sur les trigrammes en utilisant le classifieur TPCM(0,00). Le tableau se décompose en trois sous-tableaux, celui du haut est consacré aux unigrammes et est la copie conforme du tableau 7.10, celui du milieu est consacré aux bigrammes et celui du bas aux trigrammes. Chacun des sous-tableaux se lit exactement de la même manière que le tableau 7.10 page 185.

Nous avons effectué les mêmes mesures avec le classifieur TNB(0,00) et obtenu des résultats très proches.

L'information la plus notable que transmet immédiatement ce tableau est la variation de l'écart moyen de précision entre les critères qui considèrent tous les mots et ceux qui ne considèrent que les mots pleins. Cet écart est toujours en faveur des critères qui considèrent tous les mots. Mais ce qui est remarquable, c'est que cet écart passe de 4,7% pour les unigrammes à 10% pour les bigrammes et à 18% pour les trigrammes ! La juxtaposition de mots pleins ne semble pas être un indicateur viable pour la désambiguïsation [3].

Une autre information remarquable est constituée par les écarts de la précision moyenne en fonction du choix du premier paramètre. Nous observons un écart important entre les étiquettes *jeton* ou *lemme* et les étiquettes *ems* ou *smallems* pour les unigrammes. Cet écart important, de l'ordre de 16% pour les unigrammes se réduit à 13% pour les bigrammes et tombe à seulement 5% pour les trigrammes. Nous pouvons formuler deux hypothèses pour expliquer ce phénomène :

- la variabilité des trigrammes constitués d'étiquettes morphosyntaxiques est bien plus grande que celle des unigrammes et peut s'avérer suffisante pour qu'un apprentissage fiable soit réalisé ;
- un trigramme constitué d'étiquettes morphosyntaxiques peut véhiculer une information sur une construction syntaxique suffisante pour la levée de l'ambiguïté.

7.4.5 Les n-grammes doivent-il contenir le mot à désambiguïser ?

Les n-grammes doivent-ils contenir le mot à désambiguïser ou au moins lui être adjacents ? Les tableaux 7.17, 7.18, 7.20 et 7.21 montrent que la meilleure taille de contexte pour les meilleurs critères ne respecte pas cette contrainte. En d'autres termes, les n-grammes pertinents peuvent très bien, ni contenir le mot à désambiguïser, ni lui être adjacents. Par exemple, la taille de contexte optimale du critère *[2gr]-[lemme]-[differencie]-[mot]* (le critère basé sur les bigrammes qui obtient la meilleure précision) est de plus ou moins quatre mots. Ainsi, ce critère produit, entre autres, des bigrammes qui sont séparés par un ou deux mots du mot à désambiguïser. Cependant, nous ne pouvons tirer la conclusion qu'il ne faut pas se contraindre à contenir où à jouxter le mot cible sur cette seule observation. En effet, il est possible que l'éloignement des bigrammes permette d'aller chercher une information qui pourrait être capturée par l'utilisation conjointe d'un ou de plusieurs n-grammes plus grands.

Le tableau 7.24 donne les performances réalisées par :

1. le critère *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins un mot (tous les n-grammes produits contiennent le mot à désambiguïser) ;
2. le critère *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins quatre mots (certains n-grammes produits ne jouxtent pas le mot à désambiguïser) ;
3. l'utilisation conjointe des quatre critères suivants (tous les n-grammes produits contiennent le mot à désambiguïser) :
   - *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins un mot ;

3. Il faut modérer cette affirmation dans le sens où il est possible que notre corpus soit trop petit pour pouvoir réaliser un apprentissage efficace sur des bigrammes, et encore plus sur des trigrammes, composés uniquement de mots pleins, en raison de leur grande variabilité.

Critère	Contexte	TPCM(0,00)	TNB(0,00)
[2gr]-[lemme]-[differencie]-[mot]	±1 mots	65,3%	68,3%
[2gr]-[lemme]-[differencie]-[mot]	±4 mots	74,6%	76,5%
[2gr]-[lemme]-[differencie]-[mot]	±1 mots		
[3gr]-[lemme]-[differencie]-[mot]	±2 mots	74,6%	74,3%
[4gr]-[lemme]-[differencie]-[mot]	±3 mots		
[5gr]-[lemme]-[differencie]-[mot]	±4 mots		
[2gr]-[lemme]-[differencie]-[mot]	±4 mots		
[3gr]-[lemme]-[differencie]-[mot]	±4 mots	75,4%	75,9%
[4gr]-[lemme]-[differencie]-[mot]	±4 mots		
[5gr]-[lemme]-[differencie]-[mot]	±4 mots		

Tableau 7.24 – Évaluation comparative de critères basés sur les n-grammes contraints à contenir le mot cible avec des critères non assujettis à cette contrainte.

- *[3gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins deux mots ;
- *[4gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins trois mots ;
- *[5gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins quatre mots ;

4. l'utilisation conjointe des quatre critères suivants (certains n-grammes produits ne jouxtent pas le mot à désambiguïser) :
   - *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins quatre mots ;
   - *[3gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins quatre mots ;
   - *[4gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins quatre mots ;
   - *[5gr]-[lemme]-[differencie]-[mot]* avec un contexte de plus ou moins quatre mots .

Cette expérience permet de confirmer que la contrainte consistant à contenir le mot à désambiguïser, ou au moins lui être adjacent, se traduit par une baisse des performances. Il n'y a donc aucune raison d'imposer cette contrainte aux critères.

7.4.6 Pourquoi les bigrammes fonctionnent-ils mieux que les unigrammes ?

Pour illustrer la raison pour laquelle les bigrammes fonctionnent mieux que les unigrammes, nous allons nous baser sur un vocable, l'adjectif *utile* et comparer, pour ce vocable, un critère basé sur les unigrammes, *[1gr]-[lemme]-[ordonne]-[mot]* avec son équivalent basé sur les bigrammes *[2gr]-[lemme]-[ordonne]-[mot]* tous deux avec un contexte de plus ou moins trois mots. Le critère *[1gr]-[lemme]-[ordonne]-[mot]* avec un contexte de plus ou moins trois mots est celui qui obtient le meilleur résultat parmi les critères basés sur les unigrammes. Bien que son équivalent pour les bigrammes ne soit pas le meilleur, nous l'avons tout de même retenu de manière à ce que la comparaison ne soit pas biaisée par d'autres considérations que celle de la différence entre les unigrammes et les bigrammes.

Indices produits par le critère	**Lexies de l'adjectif *utile***									**Fiabi-**
[2gr]-[lemme]-[ordonne]-[mot]	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6.1**	**1.6.2**	**1.6.3**	**2**	**lité**
2gr_LOM_03_(1)_[pour_le]	0	14	0	1	0	0	0	0	1	0,72
2gr_LOM_03_(-2)_[pas_juger]	0	0	3	0	0	0	0	0	0	0,68
2gr_LOM_03_(-3)_[avoir_pas]	0	0	3	0	0	0	0	0	0	0,68
2gr_LOM_03_(-3)_[il_pouvoir]	0	0	0	3	0	0	0	0	0	0,68
2gr_LOM_03_(1)_[pour_que]	0	3	0	0	0	0	0	0	0	0,68
2gr_LOM_03_(-2)_[il_être]	0	2	0	13	0	0	0	0	2	0,64
Indices produits par le critère	**Lexies de l'adjectif *utile***									**Fiabi-**
[1gr]-[lemme]-[ordonne]-[mot]	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6.1**	**1.6.2**	**1.6.3**	**2**	**lité**
1gr_LOM_03_(-2)_il	0	4	0	15	0	0	0	0	2	0,61
1gr_LOM_03_(-3)_il	3	1	2	12	0	0	0	0	0	0,56
1gr_LOM_03_(-1)_juger	1	2	7	1	0	0	0	0	0	0,52
1gr_LOM_03_(-3)_avoir	1	1	4	0	0	0	0	0	0	0,51
1gr_LOM_03_(1)_pour	0	23	19	1	0	0	0	2	1	0,45
1gr_LOM_03_(-2)_pas	0	1	3	1	0	0	0	0	0	0,45
1gr_LOM_03_(2)_le	16	27	1	4	0	4	0	2	3	0,43
1gr_LOM_03_(-2)_pouvoir	2	2	0	3	0	0	0	0	0	0,34
1gr_LOM_03_(-1)_être	16	22	6	16	0	0	0	0	5	0,31
1gr_LOM_03_(2)_que	2	3	0	0	0	3	1	0	0	0,28

Tableau 7.25 – Les mots qui composent les bigrammes pertinents générés par le critère *[2gr]-[lemme]-[ordonne]-[mot]* ne sont pas pertinents lorsqu'ils sont générés indépendamment par le critère *[1gr]-[lemme]-[ordonne]-[mot]*. La dernière colonne précise l'indice de fiabilité tel que calculé par le classifieur TPCM(0,00).

Le critère *[1gr]-[lemme]-[ordonne]-[mot]* obtient une précision de désambiguïsation de 67,4% sur l'adjectif *utile* avec un contexte de plus ou moins trois mots et le critère *[2gr]-[lemme]-[ordonne]-[mot]* une précision de 74,7%.

En premier lieu, nous devons préciser que le critère *[1gr]-[lemme]-[ordonne]-[mot]* obtient une précision moyenne sur les 60 vocables de 65,4% pour un contexte de plus ou moins un mot. Comme nous l'avons dit dans la section précédente, puisque le critère *[2gr]-[lemme]-[ordonne]-[mot]* est une juxtaposition d'étiquettes *lemme*, pour un contexte de plus ou moins un mot ce critère véhicule exactement la même information que le critère *[1gr]-[lemme]-[ordonne]-[mot]*. Parmi les indices générés par le critère *[2gr]-[lemme]-[ordonne]-[mot]* nous trouvons donc l'unigramme de droite et celui de gauche bien que ces unigrammes soient déguisés en bigramme. Comme cette information est certainement la plus importante (cf. section 7.3.7), il est logique que ce critère basé sur les bigrammes obtienne de bonnes performances.

La différence entre ces deux critères se fait donc avec des contextes de taille supérieure. Dans de tels contextes, la présence de la juxtaposition de deux mots semble plus pertinente que la seule présence d'un mot isolé. Le tableau 7.25 illustre ce phénomène. Par exemple, l'indice *2gr_LOM_03_(1)_[pour_le]* généré par le critère basé sur les bigrammes possède une mesure de fiabilité de 0,72 alors que les deux mots qui le composent produisent, avec le critère basé sur les unigrammes, les indices *1gr_LOM_03_(1)_pour* et *1gr_LOM_03_(2)_le* pour lesquels la mesure de fiabilité n'est plus que de respectivement 0,45 et 0,43

Parfois, la présence d'un mot isolé peut suffire à lever l'ambiguïté. Mais l'information véhiculée par un tel mot n'est pas forcément perdue en ne prenant en compte que des bigrammes. Par exemple, la présence d'un nom commun isolé, quand il est perti-

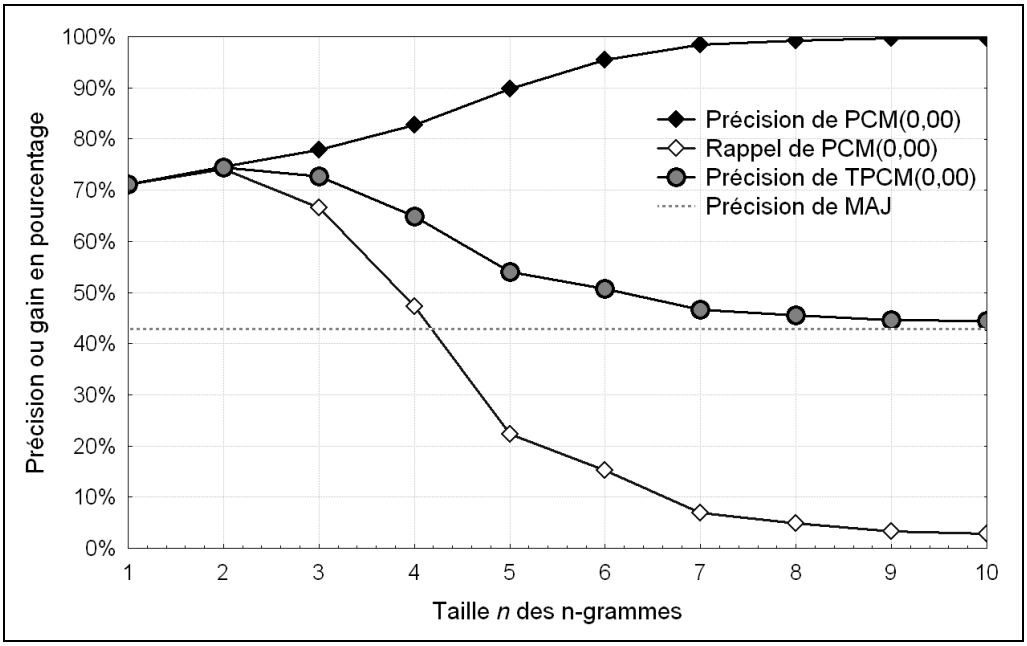


Figure 7.19 – Précision et rappel de l'algorithme PCM(0,00) et précision des algorithmes TPCM(0,00) et MAJ en fonction de la taille n des n-grammes. Le critère utilisé est *[lemme]-[differencie]-[mot]*. La taille de contexte utilisée est plus ou moins trois mots pour les 1,2,3,4,5,6-grammes, plus ou moins quatre mots pour les 8,9-grammes et plus ou moins cinq mots pour les 10-grammes.

nent, suffit souvent à lever l'ambiguïté. Or, dans ce cas, la juxtaposition ne constitue pas forcément un handicap. En effet, la plupart du temps, le nom commun est précédé d'un déterminant, d'une préposition ou d'une apostrophe. La variabilité de la lemmatisation de ce déterminant, de cette proposition ou de cette apostrophe, pour un nom commun donné situé à une distance donnée d'un vocable donné, est faible. Ainsi, l'information apportée par la juxtaposition du nom et du mot qui le précède est souvent très proche de l'information apportée par le nom seul. Prenons par exemple le vocable *détention*. L'unigramme *1gr_LOM_03_(3)_arme* est généré 13 fois pour la lexie 2 et aucune fois pour la 1 par le critère *[1gr]-[lemme]-[ordonne]-[mot]* (cf. tableau 7.7). L'information véhiculée par ce nom pertinent est entièrement capturée par le bigramme *2gr_LOM_03_(2)_['_arme]* qui est également généré 13 fois pour la lexie 2 et aucune pour la 1 par le critère *[2gr]-[lemme]-[ordonne]-[mot]*.

7.4.7 Comportement des n-grammes pour de grandes valeurs de n

Nous nous intéressons dans cette section aux n-grammes avec n allant jusqu'à 10.

La figure 7.19 donne la précision et le rappel de l'algorithme PCM(0,00) et la précision des algorithmes TPCM(0,00) et MAJ en fonction de la taille n des n-grammes. Sur ce graphique, nous observons que seuls les bigrammes et les trigrammes fonctionnent mieux que les unigrammes pour un étiquetage de toutes les occurrences (TPCM(0,00)). En revanche, au niveau du travail réellement réalisé par le classifieur (PCM(0,00)), plus la taille des n-grammes croît, plus la précision croît. La précision frôle les 100% à par-

tir des 8-grammes. Cependant, ce gain en précision se fait au détriment du rappel qui s'effondre alors en dessous de 5%.

7.5 Combiner les critères pour améliorer les résultats

7.5.1 Introduction

L'objectif de cette section est de rechercher une combinaison de critères permettant d'améliorer les performances du meilleur critère isolé. La question fondamentale n'est pas tant de trouver la combinaison optimale, que de savoir si une combinaison peut permettre une amélioration importante de la précision de la désambiguïsation (*i.e.* une amélioration qui nous rapprocherait de l'ITA de l'ordre de 96,4% [4]). Dans les sections précédentes, nous avons observé 24 familles de critères basées sur des unigrammes, 24 basées sur des bigrammes et 24 basées sur des trigrammes. Dans chaque cas, nous avons cherché à déterminer la taille optimale d'un contexte symétrique. En prenant chacun de ces critères avec sa taille optimale, nous obtenons un total de 24×3 soit 72 critères. Pour rechercher notre combinaison de critères, nous procédons de la manière suivante :

- nous sélectionnons le meilleur des 72 critères ;
- nous évaluons ce critère en conjonction avec l'un des 72 critères, et ce pour chacun de ces 72 critères [5] ;
- nous retenons la meilleure conjonction de deux critères ;
- nous évaluons cette conjonction de deux critères en ajoutant l'un des 72 critères, et ce pour chacun de ces 72 critères ;
- nous retenons la meilleure conjonction de trois critères ;
- etc.

Nous itérons ce processus jusqu'à ce que l'amélioration des performances devienne faible.

Bien entendu, cette méthode ne nous garantit absolument pas d'obtenir la combinaison optimale, mais elle devrait tout de même nous permettre d'obtenir une bonne combinaison. Cette expérience devrait également nous permettre de savoir si une combinaison des critères étudiés peut aboutir à une amélioration importante des performances par rapport aux performances des critères évalués indépendamment.

Les trois catégories grammaticales ne réagissent pas de la même manière aux critères, aussi, pour cette recherche de combinaisons, nous choisissons de les traiter indépendamment les unes des autres.

Nous évaluons nos combinaisons avec les classifieurs TPCM(0,00) et TNB(0,00). Ceux-ci ne réagissent pas de la même manière aux critères et risquent également de ne pas réagir de la même manière à leur combinaison. Nous choisissons donc de mener des études séparées pour chacun de ces classifieurs.

4. Cf. section 6.3.1 page 120

5. Un critère peut très bien être évalué avec lui-même. Ce cas de figure est peu pertinent avec le classifieur TNB(0,00) mais peut s'avérer judicieux avec le classifieur TPCM(0,00) pour donner plus de poids à un critère particulier (par exemple en utilisant deux fois un même critère en conjonction avec un critère différent utilisé une seule fois).

7.5.2 Combiner en utilisant le classifieur TPCM(0,00)

Résultats sur les noms

Le meilleur critère pour les 20 noms de notre étude, en utilisant le classifieur TPCM(0,00), est *[2gr]-[jeton]-[ordonne]-[mot]* (cf. tableau 7.17). La précision obtenue pour ce critère avec le classifieur TPCM(0,00) est de 82,03%. La meilleure combinaison de deux critères est obtenue avec le critère *[2gr]-[lemme]-[non-ordonne]-[mot-plein]* avec une amélioration des performances de 0,60%. Lors de l'ajout d'un troisième critère, le critère donnant les meilleurs résultats est *[2gr]-[jeton]-[non-ordonne]-[mot-plein]* mais l'amélioration obtenue est très faible (0,04%). Le tableau 7.26 résume les résultats de cette expérience.

Résultats sur les adjectifs

Le meilleur critère pour les 20 adjectifs de notre étude, en utilisant le classifieur TPCM(0,00), est *[2gr]-[lemme]-[ordonne]-[mot]* (cf. tableau 7.17). La précision obtenue pour ce critère avec le classifieur TPCM(0,00) est de 76,43%. La meilleure combinaison de deux critères est obtenue avec le critère *[3gr]-[lemme]-[differencie]-[mot]* avec une amélioration des performances de seulement 0,24%. Lors de l'ajout d'un troisième critère, le critère donnant les meilleurs résultats est *[3gr]-[jeton]-[ordonne]-[mot-plein]* et l'amélioration obtenue est encore plus faible : 0,13%. Le tableau 7.27 résume les résultats de cette expérience.

Résultats sur les verbes

Le meilleur critère pour les 20 verbes de notre étude, en utilisant le classifieur TPCM(0,00), est *[2gr]-[lemme]-[differencie]-[mot]* (cf. tableau 7.17). La précision obtenue pour ce critère avec le classifieur TPCM(0,00) est de 71,99%. La meilleure combinaison de deux critères est obtenue avec le critère *[3gr]-[lemme]-[differencie]-[mot]* avec une amélioration des performances de 1,06%. Lors de l'ajout d'un troisième critère, le critère donnant les meilleurs résultats est *[1gr]-[lemme]-[non-ordonne]-[mot-plein]* et l'amélioration obtenue est de 0,36%. Le tableau 7.28 résume les résultats de cette expérience.

Dans la section 7.3.7 (cf. tableau 7.15) nous avons vu qu'un contexte dissymétrique, décalé vers la droite de un mot, pouvait améliorer les résultats pour les verbes. Nous avons donc relancé le dernier traitement, celui comportant trois critères, en décalant le contexte de ces trois critères de un mot vers la droite. La précision ainsi obtenue est de 73,66% ce qui constitue une petite amélioration de 0,25% par rapport à la précision de 73,41% obtenue lorsque les contextes étaient centrés.

7.5.3 En utilisant le classifieur TNB(0,00)

Résultats sur les noms

Le meilleur critère pour les 20 noms de notre étude, en utilisant le classifieur TNB(0,00), est *[2gr]-[lemme]-[differencie]-[mot]* (cf. tableau 7.18). La précision obtenue pour ce critère avec le classifieur TNB(0,00) est de 83,59%. La combinaison de critères donne de meilleurs résultats que pour le classifieur TPCM(0,00). Nous avons ainsi combiné quatre critères avant que l'amélioration ne devienne faible. Le tableau 7.29 résume les résultats de cette expérience.

Combinaison de critères	T	Précision	Amélioration
[2gr]-[jeton]-[ordonne]-[mot]	4	82,03%	0,00%
[2gr]-[jeton]-[ordonne]-[mot] et *[2gr]-[lemme]-[non-ordonne]-[mot-plein]*	4 2	82,63%	0,60%
[2gr]-[jeton]-[ordonne]-[mot] et *[2gr]-[lemme]-[non-ordonne]-[mot-plein]* et *[2gr]-[jeton]-[non-ordonne]-[mot-plein]*	4 2 3	82,67%	0,04%

Tableau 7.26 – Combinaisons de critères pour les **noms** avec le classifieur TPCM(0,00). La première colonne précise la combinaison de critères utilisée. La seconde colonne indique la taille de la fenêtre pour chacun des critères. La troisième colonne mentionne la précision obtenue avec cette combinaison. La quatrième colonne donne l'amélioration réalisée par rapport à la combinaison précédente.

Combinaison de critères	T	Précision	Amélioration
[2gr]-[lemme]-[ordonne]-[mot]	3	76,43%	0,00%
[2gr]-[lemme]-[ordonne]-[mot] et *[3gr]-[lemme]-[differencie]-[mot]*	3 5	76,68%	0,24%
[2gr]-[lemme]-[ordonne]-[mot] et *[3gr]-[lemme]-[differencie]-[mot]* et *[3gr]-[jeton]-[ordonne]-[mot-plein]*	3 5 4	76,80%	0,13%

Tableau 7.27 – Combinaisons de critères pour les **adjectifs** avec le classifieur TPCM(0,00). Ce tableau se lit de la même manière que le tableau 7.26.

Combinaison de critères	T	Précision	Amélioration
[2gr]-[lemme]-[differencie]-[mot]	4	71,99%	0,00%
[2gr]-[lemme]-[differencie]-[mot] et *[3gr]-[lemme]-[differencie]-[mot]*	4 6	73,05%	1,06%
[2gr]-[lemme]-[differencie]-[mot] et *[3gr]-[lemme]-[differencie]-[mot]* et *[1gr]-[lemme]-[non-ordonne]-[mot-plein]*	4 6 2	73,41%	0,36%

Tableau 7.28 – Combinaisons de critères pour les **verbes** avec le classifieur TPCM(0,00). Ce tableau se lit de la même manière que le tableau 7.26.

Combinaison de critères	T	Précision	Amélioration
[2gr]-[lemme]-[differencie]-[mot]	4	83,59%	0,00%
[2gr]-[lemme]-[differencie]-[mot] et *[2gr]-[ems]-[non-ordonne]-[mot]*	4 1	84,79%	1,20%
[2gr]-[lemme]-[differencie]-[mot] et *[2gr]-[ems]-[non-ordonne]-[mot]* et *[1gr]-[lemme]-[differencie]-[mot-plein]*	4 1 1	85,69%	0,90%
[2gr]-[lemme]-[differencie]-[mot] et *[2gr]-[ems]-[non-ordonne]-[mot]* et *[1gr]-[jeton]-[differencie]-[mot-plein]* et *[1gr]-[lemme]-[non-ordonne]-[mot]*	4 1 1 2	86,02%	0,33%

Tableau 7.29 – Combinaisons de critères pour les **noms** avec le classifieur TNB(0,00). La première colonne précise la combinaison de critères utilisée. La seconde colonne indique la taille de la fenêtre pour chacun des critères. La troisième colonne mentionne la précision obtenue avec cette combinaison. La quatrième colonne donne l'amélioration réalisée par rapport à la combinaison précédente.

Combinaison de critères	T	Précision	Amélioration
[2gr]-[lemme]-[differencie]-[mot]	3	77,92%	0,00%
[2gr]-[lemme]-[differencie]-[mot] et *[2gr]-[jeton]-[non-ordonne]-[mot-plein]*	3 2	78,69%	0,77%
[2gr]-[lemme]-[differencie]-[mot] et *[2gr]-[jeton]-[non-ordonne]-[mot-plein]* et *[2gr]-[ems]-[differencie]-[mot-plein]*	3 2 1	78,95%	0,26%

Tableau 7.30 – Combinaisons de critères pour les **adjectifs** avec le classifieur TNB(0,00). Ce tableau se lit de la même manière que le tableau 7.29.

Combinaison de critères	T	Précision	Amélioration
[2gr]-[lemme]-[differencie]-[mot]	4	74,00%	0,00%
[2gr]-[lemme]-[differencie]-[mot] et *[1gr]-[ems]-[ordonne]-[mot]*	4 2	75,93%	1,92%
[2gr]-[lemme]-[differencie]-[mot] et *[1gr]-[ems]-[ordonne]-[mot]* et *[1gr]-[lemme]-[non-ordonne]-[mot-plein]*	4 2 1	76,87%	0,95%
[2gr]-[lemme]-[differencie]-[mot] et *[1gr]-[ems]-[ordonne]-[mot]* et *[1gr]-[lemme]-[non-ordonne]-[mot-plein]* et *[2gr]-[jeton]-[ordonne]-[mot]*	4 2 1 4	77,19%	0,32%

Tableau 7.31 – Combinaisons de critères pour les **verbes** avec le classifieur TNB(0,00). Ce tableau se lit de la même manière que le tableau 7.29.

Le classifieur TNB(0,00) donne de meilleurs résultats que le classifieur TPCM(0,00) et semble tirer bien mieux parti de la combinaison et de la complémentarité des critères. Dans la section 2.7.2, nous avons dressé un inventaire non exhaustif des sources d'information utiles et utilisées pour la désambiguïsation lexicale. Au niveau des co-occurrences, nous avons distingué les informations locales et les informations globales. Cependant, la meilleure combinaison de quatre critères que nous avons obtenue n'utilise que des informations locales. Nous pourrions peut être améliorer la précision en essayant de capturer une information globale avec l'un de ces quatre critères, les trois autres se chargeant de l'information locale mais aussi de minimiser le bruit généré par le quatrième critère. Dans le dernier paragraphe de la section 7.3.5 nous avons observé qu'en s'éloignant de la cible, le gain tend vers zéro de manière bien moins rapide pour les noms que pour les adjectifs et les verbes. La catégorie grammaticale des noms semble donc être celle qui se prête le mieux à la capture d'une information globale. Logiquement, les indices les plus pertinents en s'éloignant de la cible sont des jetons ou des lemmes (pas des étiquettes morphosyntaxiques) de mots pleins (les mots grammaticaux n'apportent probablement rien lorsqu'ils sont éloignés de la cible) sans tenir compte de leur position par rapport au mot à désambiguïser (*i.e.* en utilisant le contexte *non-ordonne* ou *differencie*). Des quatre critères que nous avons retenus pour une utilisation conjointe, celui qui correspond le mieux à ce cahier des charges est *[1gr]-[jeton]-[differencie]-[mot-plein]*. Pour ce critère utilisé isolément, la taille de contexte optimale était plus ou moins un mot. Dans le cadre d'une utilisation conjointe avec d'autres critères, il est possible que nous gagnions en précision en élargissant ce contexte. En augmentant progressivement la taille du contexte, nous avons obtenue la meilleure précision pour une taille de seulement plus ou moins trois *mot-plein*. Notre tentative de capturer une information globale est donc un échec. Le gain obtenu par rapport au critère est tout de même de 0,47% et la précision atteinte est de 86,49% en utilisant conjointement les critères :

- *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de ±4 mots ;
- *[2gr]-[ems]-[non-ordonne]-[mot]* avec un contexte de ±1 mot ;
- *[1gr]-[jeton]-[differencie]-[mot-plein]* avec un contexte de ±3 mots pleins ;
- *[1gr]-[lemme]-[non-ordonne]-[mot]* avec un contexte de ±2 mots.

Notre échec pour capturer une information globale n'est pas surprenant. Les trois critères capturant l'information locale génèrent au maximum $(4 + 1 + 2) \times 2 = 14$ indices. En prenant un contexte susceptible de capturer une information globale, par exemple ±50 mots, c'est une centaine d'indices qui sont générés et les 14 indices véhiculant l'information locale ne peuvent faire face au bruit généré par une telle quantité d'indices. Nous discutons de ce problème dans le dernier chapitre section 8.3.

Résultats sur les adjectifs

Le meilleur critère pour les 20 adjectifs de notre étude, en utilisant le classifieur TNB(0,00), est *[2gr]-[lemme]-[differencie]-[mot]* (cf. tableau 7.18). La précision obtenue pour ce critère avec le classifieur TNB(0,00) est de 77,92%. La meilleure combinaison de deux critères est obtenue avec le critère *[2gr]-[jeton]-[non-ordonne]-[mot-plein]* avec une amélioration des performances de 0,77%. Lors de l'ajout d'un troisième critère, le critère donnant les meilleurs résultats est *[2gr]-[ems]-[differencie]-[mot-plein]* et l'amélioration obtenue n'est plus que de 0,26%. Le tableau 7.30 résume les résultats de cette expérience.

Résultats sur les verbes

Le meilleur critère pour les 20 verbes de notre étude, en utilisant le classifieur TNB(0,00), est *[2gr]-[lemme]-[differencie]-[mot]* (cf. tableau 7.18). La précision obtenue pour ce critère avec le classifieur TNB(0,00) est de 74,00%. La combinaison de critères donne de meilleurs résultats que pour le classifieur TPCM(0,00). Nous avons ainsi combiné quatre critères avant que l'amélioration ne devienne faible. Le tableau 7.31 résume les résultats de cette expérience.

Dans la section 7.3.7 (cf. tableau 7.15) nous avons vu qu'un contexte dissymétrique, décalé vers la droite de un mot, pouvait améliorer les résultats pour les verbes. Nous avons donc relancé le dernier traitement, celui comportant quatre critères, en décalant le contexte de ces quatre critères de un mot vers la droite. La précision ainsi obtenue est de 77,92% ce qui constitue une amélioration substantielle de 0,73% par rapport à la précision de 77,19% obtenue lorsque les contextes étaient centrés.

7.6 Synthèse des résultats

7.6.1 Critères évalués indépendamment

Les approches basées sur corpus étiquetés fondent leur désambiguïsation sur des connaissances tirées du contexte des mots à désambiguïser. Les questions concernant la taille ou les limitations de ce contexte sont donc de première importance. D'autres questions se posent également comme :

- Quels sont les mots du contexte à considérer (faut-il, par exemple, éliminer les mots grammaticaux) ?
- Quelles sont les étiquettes (forme brute, lemme, étiquette morphosyntaxique, etc.) des mots du contexte utiles pour la désambiguïsation ?

Dans les sections 7.3 et 7.4 nous avons effectué une étude que nous espérons rigoureuse, systématique et relativement complète, des critères basés sur les n-grammes, et plus particulièrement les unigrammes (*i.e.* cooccurrences), les bigrammes et les trigrammes, évalués indépendamment pour tenter d'apporter des réponses à toutes ces questions.

Faut-il accepter les mots en dehors de la phrase ?

Nous avons choisi de limiter le contexte aux seuls mots de la phrase. Les ponctuations fortes de fin de phrase sont comprises dans le contexte. Les n-grammes peuvent donc très bien contenir des ponctuations faibles. Les mots en dehors de la phrase sont toujours ignorés, et ce pour tous les n-grammes. Cette politique, qui semble être celle qui donne les meilleurs résultats (cf. section 7.3.2 pour les unigrammes et section 7.4.2 pour les bigrammes et les trigrammes), ne constitue pas un élément primordial car, d'après nos expériences, elle n'apporte qu'une faible amélioration de l'ordre de 0,3% de la précision de la désambiguïsation.

Faut-il incorporer le mot à désambiguïser ?

La forme brute du vocable à désambiguïser est variable en fonction de ses occurrences. Le mot peut contenir une ou plusieurs majuscules, et son orthographe peut varier en fonction de son nombre, de son genre, voire de son temps, son mode et sa personne pour les verbes. L'étiquette morphosyntaxique est également variable. Même le lemme, qui est unique pour un vocable donné, apporte une information sur la fréquence

relative des lexies quand il est systématiquement retourné parmi les indices générés par un critère. La question de l'intérêt de l'incorporation du mot à désambiguïser parmi les mots considérés est donc légitime.

Dans le cas des unigrammes, l'amélioration ou la dégradation de la précision de la désambiguïsation apportée par la prise en compte du mot à désambiguïser est fonction de l'algorithme de classification utilisé et de l'étiquette des mots retenue (cf. section 7.3.3). Dans tous les cas, lorsque le critère retourne l'étiquette morphosyntaxique (*ems*) des mots, il semble qu'il y ait une amélioration. Cette amélioration peut atteindre 1,4% avec le classifieur `TNB(0,00)`. Dans le cadre de notre étude et pour les unigrammes, nous avons choisi de n'incorporer le mot à désambiguïser que pour les unigrammes constitués de l'étiquette morphosyntaxique du mot, c'est-à-dire pour les critères de la forme *[ems]-[<param2>]-[<param3>]*.

Dans le cas des n-grammes avec $n > 1$, le problème est légèrement différent. En effet, aucun indice ne peut se réduire à la seule information du mot cible puisque les indices sont au moins constitués de deux mots. Nous avons choisi de toujours incorporer le mot cible de manière à ce que tous les indices retournés par un critère basé sur des n-grammes soient bien composés de la juxtaposition de n mots.

La lemmatisation améliore-t-elle la désambiguïsation ?

Il est clair que ce sont les étiquettes *jeton* et *lemme* qui permettent d'obtenir la désambiguïsation la plus fiable. La lemmatisation n'apporte pas, en moyenne, d'amélioration importante des performances pour les unigrammes (cf. tableau 7.10 et 7.11 pages 185). L'amélioration de la précision du meilleur critère basé sur les unigrammes et les lemmes est tout de même de 0,7% par rapport à la précision du meilleur critère basé sur les unigrammes et les jetons (cf. tableau 7.5 page 175). Lorsque la taille des n-grammes croît, l'amélioration moyenne de la précision due à la lemmatisation semble augmenter et est de 1,5% pour les bigrammes et de 3,4% pour les trigrammes (cf. tableau 7.23 page 208). L'amélioration de la précision du meilleur critère basé sur les lemmes par rapport à la précision du meilleur critère basé sur les jetons est de 1,1% pour les bigrammes (cf. tableau 7.18 page 202) et de 4,2% pour les trigrammes (cf. tableau 7.21 page 205). La lemmatisation améliore donc la précision de la désambiguïsation de manière substantielle mais elle ne semble pas primordiale dans le cadre des unigrammes.

Nous tenons à remarquer que la lemmatisation est essentiellement une réponse partielle au problème de la dispersion des données (cf. section 2.5.6). La dispersion des données augmentant avec la taille des n-grammes, il est logique que l'amélioration qu'elle apporte augmente également avec la taille des n-grammes.

La lemmatisation apporte autre chose qu'une simple réponse au problème de la dispersion des données, elle permet de lever partiellement ou totalement l'ambiguïté catégorielle sur les mots. Par exemple, le mot *porte* peut correspondre à un nom commun (porte), un verbe (porter) ou un adjectif (porte). L'étiquette *jeton* souffre donc de ce problème d'ambiguïté catégorielle qui ne facilite pas la désambiguïsation lexicale. Pour lever cette ambiguïté catégorielle, nous accolons l'étiquette *ems* à tous les indices de type *jeton* générés par des critères basés sur des unigrammes. Le gain obtenu par cette juxtaposition est de l'ordre de 0,4% (cf. section 7.3.4). Nous n'effectuons pas cette juxtaposition pour les critères basés sur des n-grammes avec $n > 1$ car elle n'apporte rien puisque, dans la plupart des cas, les mots contigus incorporés au n-gramme permettent de lever l'ambiguïté catégorielle (cf. section 7.4.3).

Faut-il filtrer les mots à prendre en considération ?

De nombreuses études (Mooney, 1996 ; El-Bèze *et al.*, 1998, 1999 ; Ng & Lee, 2002 ; etc.) ne considèrent pas tous les mots du contexte. Généralement les mots grammaticaux sont ignorés. Dans nos expériences, nous avons évalué des critères qui ne retiennent que les mots pleins, c'est-à-dire les noms, les adjectifs, les verbes et les adverbes, et des critères qui prennent en compte tous les mots y compris les mots grammaticaux. Nous avons toujours observé une dégradation de la précision de la désambiguïsation lorsque les mots grammaticaux sont retirés. Cette dégradation, en moyenne, pour les critères basés sur des unigrammes, est de l'ordre de 0,3% pour les noms, 2,5% pour les adjectifs et 6,9% pour les verbes (cf. tableau 7.10 page 185). Cette dégradation devient respectivement pour les noms, les adjectifs et les verbes, de 2,7%, 3,4% et 13,5% pour les critères basés sur des bigrammes, et de 12,4%, 15,9% et 20,2% pour les critères basés sur des trigrammes (cf. tableau 7.10). Ces résultats sont ceux obtenus avec le classifieur TPCM(0,00). En utilisant le classifieur TNB(0,00) la dégradation engendrée par le retrait des mots grammaticaux semble encore plus importante (cf. tableau 7.10 *versus* tableau 7.11 pages 185).

Dans la section 7.3.7 nous proposons un critère qui ne considère que les mots dont la catégorie grammaticale donne de bons résultats pour la désambiguïsation. Il s'agit d'un critère qui opère un filtrage bien plus pertinent que de ne retenir que les noms, les adjectifs, les verbes et les adverbes. La précision atteinte en utilisant ce critère avec le classifieur TPCM(0,00) (respectivement le classifieur TNB(0,00)) est en retrait par rapport au critère équivalent considérant tous les mots de 1,2% (respectivement 2,3%) pour les noms, 3,7% (respectivement 4,2%) pour les adjectifs et 2,8% (respectivement 4,7%) pour les verbes (cf. tableau 7.13 page 190).

Il semble apparaître à l'issue de toutes ces expériences que tous les mots, quelle que soit leur classe grammaticale, participent à la levée de l'ambiguïté.

L'ordre des mots est-il important ?

La position relative de chacun des mots par rapport au mot à désambiguïser semble être importante pour les critères basés sur les unigrammes. En moyenne, la précision atteinte en utilisant des critères qui tiennent compte de la position de chacun des mots est meilleure que celle des critères équivalents qui ne tiennent pas compte de la position des mots. L'amélioration observée, en utilisant le classifieur TPCM(0,00) (respectivement le classifieur TNB(0,00)), est de 2,9% (respectivement 2,0%) pour les noms, 1,1% (respectivement 1,0%) pour les adjectifs et 2,2% (respectivement 1,5%) pour les verbes (cf. tableau 7.10 et 7.11 page 185).

En revanche, pour les critères basés sur les bigrammes et les trigrammes, tenir compte de la position de chacun des mots a plutôt tendance à dégrader les résultats. Dans ce cas, la politique optimale semble être de simplement différencier les indices du contexte gauche, les indices qui contiennent le mot à désambiguïser et les indices du contexte droit. Nous observons alors une amélioration de l'ordre de 0,5% par rapport à des critères ne tenant pas compte de l'ordre des mots (cf. tableau 7.23 page 208).

Quelle est la taille optimale du contexte ?

Nous observons que les meilleures performances sont atteintes pour de petits contextes de l'ordre de plus ou moins un à quatre mots. La taille optimale est fonction du critère utilisé, de la catégorie grammaticale du vocable à désambiguïser, et croît avec la taille des n-grammes. Pour les unigrammes, la taille optimale moyenne est de

1,5 mot pour les noms, 1,1 mot pour les adjectifs et 1,6 mot pour les verbes. Pour les bigrammes, elle est de 2,4 mots pour les noms, 2 mots pour les adjectifs et 2,6 mots pour les verbes. Pour les trigrammes, elle est de 3,1 mots pour les noms, 3,3 mots pour les adjectifs et 3,8 mots pour les verbes.

Les n-grammes (avec $n > 1$) doivent-ils forcément contenir ou jouxter le mot à désambiguïser ?

Les tailles optimales des contextes pour des critères basés sur des n-grammes avec $n > 1$ sont régulièrement suffisamment grandes pour que les indices de ces contextes ne contiennent ou ne jouxtent pas forcément le mot à désambiguïser (cf. tableaux 7.17, 7.18, 7.20 et 7.21).

D'autre part, comme nous le montrons dans la section 7.4.5, l'information véhiculée par les indices qui ne contiennent ou ne jouxtent pas le mot à désambiguïser n'est absolument pas capturée en utilisant conjointement des critères basés sur des n-grammes plus grands. Il semble donc que les n-grammes (avec $n > 1$) ne doivent pas forcément contenir ou jouxter le mot à désambiguïser.

Les contextes doivent-ils être symétriques ?

Les figures 7.12 (page 190) et 7.13 (page 191) montrent que les principaux indices utilisés pour lever l'ambiguïté des noms et des adjectifs se répartissent de manière approximativement symétrique autour du mot à désambiguïser. Il n'en va pas de même pour les verbes pour lesquels les indices semblent se situer majoritairement en aval du verbe à désambiguïser comme le montre la figure 7.14 (page 191). Pour cette catégorie grammaticale, un contexte dissymétrique décalé de un mot vers la droite apparaît plus adéquat (cf. tableau 7.15 page 193). Lors de plusieurs expériences, l'utilisation de ce contexte décalé a permis d'améliorer la précision de la désambiguïsation des verbes de 0,25% à 0,75% (cf. section 7.3.7, 7.5.2 et 7.5.3).

Quel est le critère évalué indépendamment donnant les meilleurs résultats ?

Lors de nos expériences, le classifieur `TNB(0,00)` a systématiquement surpassé le classifieur `TPCM(0,00)`.

En utilisant le classifieur `TNB(0,00)`, le critère basé sur les unigrammes permettant d'atteindre la meilleure précision est *[lemme]-[ordonne]-[mot]*. Ce critère considère le lemme de tous les mots du contexte en tenant compte de leur position. Il atteint une précision de désambiguïsation de 81,9% avec une fenêtre de plus ou moins deux mots pour les noms, de 76,8% avec une fenêtre de plus ou moins un mots pour les adjectifs, et de 71,8% avec une fenêtre de plus ou moins trois mots pour les verbes (cf. tableau 7.5 page 175).

Le critère basé sur les bigrammes permettant d'atteindre la meilleure précision est *[lemme]-[differencie]-[mot]*. Ce critère considère le lemme de tous les mots du contexte en différenciant les bigrammes selon qu'ils se trouvent à gauche, à droite ou qu'ils contiennent le mot à désambiguïser. Ce critère atteint une précision de désambiguïsation de 83,6% avec une fenêtre de plus ou moins quatre mots pour les noms, de 77,9% avec une fenêtre de plus ou moins trois mots pour les adjectifs, et de 74,0% avec une fenêtre de plus ou moins quatre mots pour les verbes (cf. tableau 7.18 page 202).

Le critère basé sur les trigrammes permettant d'atteindre la meilleure précision est également *[lemme]-[differencie]-[mot]*. Il atteint une précision de désambiguïsation de 82,3% avec une fenêtre de plus ou moins cinq mots pour les noms, de 72,7% avec une

		MAJ	PCM(0,00)	PCM(0,57)	NB(0,00)	NB(0,98)	PEBLS(5)
Noms	Précision	57,3%	81,5%	86,2%	83,7%	92,8%	80,2%
	Rappel	57,3%	81,3%	74,1%	83,4%	69,3%	80,2%
	Gain	0,0%	56,7%	67,6%	61,8%	83,0%	53,7%
	Performance	0,00	0,67	0,71	0,71	0,76	0,64
	Précision totale	57,3%	81,5%	80,1%	83,6%	81,8%	80,2%
	Gain total	0,0%	56,6%	53,5%	61,6%	57,4%	53,7%
Adjectifs	Précision	46,4%	76,6%	86,5%	78,2%	91,8%	74,7%
	Rappel	46,4%	75,6%	64,1%	77,2%	58,7%	74,7%
	Gain	0,0%	56,3%	74,7%	59,3%	84,7%	52,7%
	Performance	0,00	0,65	0,69	0,67	0,69	0,62
	Précision totale	46,4%	76,3%	71,5%	77,9%	72,4%	74,7%
	Gain total	0,0%	55,8%	46,7%	58,8%	48,4%	52,7%
Verbes	Précision	37,2%	72,1%	84,3%	74,1%	88,1%	60,9%
	Rappel	37,2%	71,9%	59,7%	74,0%	59,9%	60,9%
	Gain	0,0%	55,5%	75,0%	58,7%	81,0%	37,7%
	Performance	0,00	0,63	0,66	0,65	0,69	0,47
	Précision totale	37,2%	72,0%	65,9%	74,0%	68,0%	60,9%
	Gain total	0,0%	55,4%	45,7%	58,6%	49,0%	37,7%

Tableau 7.32 – Performances, pour les trois catégories grammaticales, obtenues avec le meilleur critère évalué indépendamment par chacun des classifieurs. Le critère utilisé est *[2gr]-[lemme]-[differencie]-[mot]* et la taille de contexte retenue est de plus ou moins quatre mots pour les noms et les verbes et plus ou moins trois mots pour les adjectifs. Les lignes *Précision totale* et *Gain total* donnent la précision et le gain obtenus sur la totalité des instances à classer en affectant à toutes les instances non classées la classe majoritaire.

fenêtre de plus ou moins trois mots pour les adjectifs, et de 71,2% avec une fenêtre de plus ou moins cinq mots pour les verbes (cf. tableau 7.21 page 205).

Le critère évalué indépendamment donnant les meilleurs résultats est donc clairement un critère basé sur les bigrammes et non un critère basé sur les unigrammes, contrairement à ce à quoi nous nous attendions.

Performances atteintes par plusieurs algorithmes de classification

Le critère évalué indépendamment qui permet d'obtenir les meilleures performances est *[2gr]-[lemme]-[differencie]-[mot]*, utilisé en conjonction avec le classifieur `TNB(0,00)`. Les tailles de contexte optimales pour ce critère et ce classifieur sont de plus ou moins quatre mots pour les noms et les verbes, et plus ou moins trois mots pour les adjectifs. Le tableau 7.32 donne les performances, pour les trois catégories grammaticales, obtenues avec ce critère par chacun des classifieurs retenus dans la section 6.14.

Les performances du classifieur `PEBLS(5)` sont en net retrait par rapport à celles qu'il obtenait dans la section 6.14. Il est possible que le critère *[2gr]-[lemme]-[differencie]-[mot]* ne le favorise pas. La précision obtenue par le classifieur `NB(0,00)` sur l'ensemble des occurrences (*i.e.* la précision du classifieur `TNB(0,00)`) est relativement bonne et de 83,6% pour les noms, 77,9% pour les adjectifs et 74,0% pour les verbes. Sans chercher à réaliser une classification sur l'ensemble des occurrences, le classifieur `NB(0,98)` obtient une précision exceptionnelle de 92,8% pour les noms, 91,8% pour les adjectifs et 88,1% pour les verbes avec un rappel respectable de respective-

ment 69,3%, 58,7% et 59,9%. Cette précision est d'autant plus surprenante qu'elle est obtenue sans chercher à optimiser spécifiquement pour cela le choix du critère, la taille de la fenêtre, les paramètres du classifieur `NB(0,98)` et sans combinaison de plusieurs critères.

Détail des performances en fonction des vocables

Le tableau 7.33 détaille pour chacun des vocables de notre étude la performance obtenue avec le meilleur critère évalué indépendamment. Les vocables sont classés par catégorie grammaticale (noms, adjectifs puis verbes) puis par ordre de précision de l'algorithme `TNB(0,00)` décroissante. À la lecture de ce tableau, nous remarquons que le gain par rapport à l'algorithme majoritaire est toujours positif et supérieur à 16,3% lorsque toutes les occurrences sont étiquetées. Nous remarquons également que certains vocables *a priori* difficiles à désambiguïser comme *pied* avec 62 lexies et une perplexité de 11,7, ou *mettre* avec 140 lexies et une perplexité de 12,7, sont plutôt bien désambiguïsés et obtiennent, lorsque toutes les occurrences sont étiquetées, une précision de respectivement 83,3% et 81,2% et un gain de respectivement 73,3% et 67,4%.

Nous avons demandé à Delphine Reymond, qui s'est acquittée de l'étiquetage lexical du corpus, de citer les vocables sur lesquels elle a rencontré le plus de difficulté pour dresser la liste des lexies (*i.e.* les vocables pour lesquels les lexies sont les plus proches et difficiles à distinguer). Contrairement à ce que nous attendions, nous n'avons observé aucune corrélation entre cette liste et les performances réalisées par l'algorithme `TNB(0,00)`. Cependant, nous pouvons remarquer que les occurrences des vocables sont désambiguïsées par l'annotateur en utilisant des critères distributionnels stricts tels que les restrictions de sélection et tels que des tests de commutation de synonymes stricts (nous pouvons remplacer « faire barrage à l'opposition » par « faire *obstacle* à l'opposition », mais nous ne pouvons pas avoir de façon naturelle « construire un *obstacle* sur un fleuve »). Or, si ces tests permettent à l'annotateur d'effectuer des distinctions entre les lexies de manière fiable, ils sont généralement inaccessibles aux critères que nous avons étudiés.

Nous observons une légère corrélation entre la précision réalisée par l'algorithme `TNB(0,00)` et le nombre de lexies des vocables. La corrélation est encore plus marquée avec la mesure de perplexité de la répartition des lexies ou encore avec la précision de l'algorithme majoritaire (`MAJ`). Les vocables qui obtiennent la moins bonne précision de désambiguïsation ont en moyenne une mesure de perplexité plus importante et une précision de l'algorithme majoritaire plus faible. Il ne s'agit que de tendances autour desquelles les fluctuations sont importantes. Il serait intéressant de réaliser une étude sur les vocables les moins bien désambiguïsés pour essayer de comprendre la raison de l'échec partiel de leur désambiguïsation. Il est possible que cet échec soit dû au fait que ces vocables sont essentiellement désambiguïsés en utilisant des critères distributionnels inaccessibles aux critères basés sur les n-grammes que nous avons mis en œuvre. Nous devons toutefois remarquer que la précision réalisée par l'algorithme `NB(0,98)` n'est jamais dramatique, ne descend jamais en dessous de 76,4% et se trouve dans la majorité des cas au-dessus de 80%. Sur les vocables difficiles, la dégradation se répercute essentiellement au niveau du rappel. Ce comportement est rassurant et correspond tout à fait à ce que nous attendons d'un tel classifieur : ne prendre une décision que quand la probabilité d'erreur est faible.

Dans la section qui suit (section 7.6.2) ainsi que dans la section 8.3 nous abordons des pistes permettant d'envisager une amélioration de la précision de la désambiguïsation.

Vocables	Fréquence	Lexie plus fréquente	Nombre de lexies	Entropie	Perplexité	Précision MAJ	Précision NB(0,98)	Rappel NB(0,98)	Précision TNB(0,00)	Gain TNB(0,00)
chef	1133	861	11	1,47	2,77	76,0%	98,7%	89,8%	96,6%	86,0%
détention	112	81	2	0,85	1,80	72,3%	97,9%	84,8%	96,4%	87,1%
solution	880	821	4	0,44	1,36	93,3%	97,9%	89,4%	96,4%	45,8%
barrage	92	70	5	1,18	2,26	76,1%	94,7%	78,3%	91,3%	63,6%
observation	572	492	3	0,68	1,60	86,0%	92,2%	74,1%	88,3%	16,3%
compagnie	412	294	12	1,62	3,08	71,4%	94,8%	75,0%	86,9%	54,2%
formation	1528	975	9	1,66	3,17	63,8%	92,4%	71,3%	85,7%	60,6%
lancement	138	110	5	0,99	1,99	79,7%	89,1%	59,4%	85,5%	28,6%
constitution	422	211	6	1,64	3,13	50,0%	93,1%	66,8%	84,6%	69,2%
pied	960	361	62	3,55	11,70	37,6%	93,1%	71,7%	83,3%	73,3%
communication	1703	691	13	2,44	5,42	40,6%	90,7%	67,9%	81,3%	68,6%
économie	930	457	10	2,16	4,46	49,1%	92,2%	61,9%	80,6%	61,9%
degré	507	297	18	2,47	5,53	58,6%	90,3%	64,3%	80,5%	52,9%
suspension	110	68	5	1,50	2,82	61,8%	93,2%	61,8%	79,1%	45,2%
restauration	104	45	5	1,85	3,60	43,3%	90,4%	45,2%	78,8%	62,7%
vol	278	112	10	2,20	4,61	40,3%	89,2%	56,5%	74,5%	57,2%
station	266	85	8	2,58	5,98	32,0%	88,1%	53,0%	70,3%	56,4%
passage	600	222	19	2,70	6,48	37,0%	84,8%	45,7%	69,3%	51,3%
concentration	246	111	6	1,98	3,93	45,1%	88,9%	52,0%	67,9%	41,5%
organe	366	140	6	2,24	4,71	38,3%	79,4%	41,0%	60,9%	36,7%
courant	170	153	4	0,62	1,54	90,0%	99,3%	80,6%	97,6%	76,5%
biologique	475	427	4	0,55	1,46	89,9%	98,1%	85,7%	95,6%	56,3%
traditionnel	447	400	2	0,49	1,40	89,5%	97,0%	73,2%	95,3%	55,3%
historique	620	542	3	0,67	1,59	87,4%	96,2%	74,4%	92,6%	41,0%
simple	1051	434	14	2,14	4,41	41,3%	93,5%	66,1%	84,8%	74,1%
sûr	645	296	14	2,61	6,12	45,9%	92,9%	73,2%	83,9%	70,2%
secondaire	195	105	5	1,69	3,23	53,8%	92,2%	66,7%	81,0%	58,9%
populaire	457	219	5	2,02	4,05	47,9%	94,8%	59,7%	79,6%	60,9%
strict	220	100	9	2,23	4,69	45,5%	92,1%	58,6%	79,1%	61,7%
utile	359	154	9	2,39	5,23	42,9%	87,3%	47,9%	77,4%	60,5%
haut	1017	254	29	3,46	10,97	25,0%	92,8%	60,0%	77,0%	69,3%
plein	844	144	35	3,99	15,93	17,1%	87,1%	49,4%	71,1%	65,1%
sensible	425	127	11	2,63	6,19	29,9%	88,6%	49,4%	69,6%	56,7%
correct	116	62	5	1,81	3,50	53,4%	84,5%	42,2%	67,2%	29,6%
sain	129	52	10	2,45	5,46	40,3%	90,0%	41,9%	65,9%	42,9%
exceptionnel	226	120	3	1,45	2,73	53,1%	77,9%	35,8%	65,5%	26,4%
clair	557	163	20	3,10	8,57	29,3%	84,8%	44,0%	62,7%	47,2%
vaste	368	156	6	2,08	4,22	42,4%	78,2%	28,3%	56,5%	24,5%
frais	184	67	18	3,12	8,70	36,4%	81,4%	38,0%	55,4%	29,9%
régulier	181	59	11	2,54	5,82	32,6%	87,3%	30,4%	51,4%	27,9%

Suite sur la page suivante...

Suite de la page précédente...

Vocables	Fréquence	Lexie plus fréquente	Nombre de lexies	Entropie	Perplexité	Précision MAJ	Précision NB(0,98)	Rappel NB(0,98)	Précision TNB(0,00)	Gain TNB(0,00)
répondre	2529	1981	9	0,99	1,99	78,3%	94,1%	79,4%	88,5%	46,9%
importer	576	159	8	2,57	5,93	27,6%	92,9%	77,1%	86,8%	81,8%
exercer	698	415	8	1,52	2,88	59,5%	92,0%	72,6%	85,2%	63,6%
conclure	727	331	16	2,36	5,13	45,5%	90,8%	66,7%	81,4%	65,9%
rendre	1990	923	27	2,88	7,35	46,4%	92,6%	67,4%	81,4%	65,2%
mettre	5246	2216	140	3,67	12,71	42,2%	92,5%	73,3%	81,2%	67,4%
comprendre	2145	700	13	2,77	6,84	32,6%	89,1%	67,8%	80,0%	70,3%
parvenir	654	240	8	2,31	4,96	36,7%	89,4%	60,7%	75,7%	61,6%
présenter	2142	859	18	2,56	5,89	40,1%	86,3%	57,1%	72,2%	53,5%
connaître	1635	655	16	2,24	4,71	40,1%	83,4%	51,1%	72,0%	53,3%
venir	3797	947	33	3,22	9,29	24,9%	85,8%	55,5%	71,5%	62,1%
poursuivre	978	354	16	2,71	6,53	36,2%	84,4%	52,7%	70,8%	54,2%
conduire	1093	417	15	2,27	4,83	38,2%	83,1%	50,4%	68,6%	49,3%
entrer	1258	335	39	3,68	12,80	26,6%	84,3%	48,8%	65,7%	53,2%
passer	2556	405	84	4,49	22,50	15,8%	83,5%	48,1%	64,6%	58,0%
porter	2347	690	59	4,02	16,27	29,4%	84,5%	50,7%	64,1%	49,1%
arrêter	916	219	15	2,97	7,83	23,9%	76,4%	40,6%	62,7%	50,9%
tirer	1002	290	47	3,88	14,77	28,9%	84,2%	46,8%	62,5%	47,2%
ouvrir	919	239	41	3,80	13,92	26,0%	80,7%	44,7%	62,5%	49,3%
couvrir	543	181	22	3,27	9,65	33,3%	80,4%	40,9%	60,2%	40,3%
Maximum	5246	2216	140	4,49	22,50	93,3%	99,3%	89,8%	97,6%	87,1%
Minimum	92	45	2	0,44	1,36	15,8%	76,4%	28,3%	51,4%	16,3%

Tableau 7.33 – Performances obtenues, avec le meilleur critère évalué indépendamment, pour chacun des vocables. Le critère utilisé est *[2gr]-[lemme]-[differencie]-[mot]* et la taille de contexte retenue est de plus ou moins quatre mots pour les noms et les verbes et de plus ou moins trois mots pour les adjectifs. Pour chaque vocable, ce tableau précise la fréquence des occurrences (colonne *Fréquence*), la fréquence de la lexie la plus fréquente (colonne *Lexie plus fréquente*), le nombre de lexies (colonne *Nombre de lexies*), l'entropie de la répartition des occurrences sur les lexies (colonne *Entropie*) et, à entropie égale, le nombre de lexies qu'aurait le vocable si ses lexies étaient équiprobables (*i.e.* la perplexité colonne *Perplexité*), la précision de l'algorithme majoritaire (colonne *Précision* `MAJ`), la précision de l'algorithme `NB(0,98)` (colonne *Précision* `NB(0,98)`) ainsi que son rappel (colonne *Rappel* `NB(0,98)`), et enfin la précision de l'algorithme `TNB(0,00)` (colonne *Précision* `TNB(0,00)`) ainsi que son gain (colonne *Gain* `TNB(0,00)`). Les deux dernières lignes précisent les valeurs maximum et minimum de chacune des colonnes.

7.6.2 Combinaisons de critères

Tout ce que nous avons dit dans la section précédente (section 7.6.1) est valable lorsqu'un seul critère est utilisé. Cette section permet donc de se faire une opinion sur les critères évalués indépendamment. Pour améliorer la précision de la désambiguïsation en combinant des critères, il serait très maladroit de sélectionner les deux ou trois critères qui obtiennent les meilleures précisions évalués indépendamment. En effet, de tels critères seraient probablement très proches et leur combinaison n'apporterait rien.

Dans la section 7.5 nous tentons de combiner des critères avec pour intention d'améliorer de manière importante la précision de la désambiguïsation. Pour cela, nous partons du critère obtenant les meilleurs résultats et essayons de le combiner systématiquement avec chacun des autres critères évalués au cours de nos expériences sur les unigrammes, les bigrammes et les trigrammes. Nous réitérons l'opération sur la meilleure combinaison obtenue jusqu'à ce que l'amélioration des performances devienne faible. Nous nous attendions à ce que le classifieur naïf de Bayes (par exemple `TNB(0,00)`) plafonne assez vite en raison de la condition d'indépendance des attributs violée de manière plus importante en combinant les critères. Nous pensions que notre liste de décisions (par exemple `TPCM(0,00)`) tirerait mieux parti de la combinaison des critères puisque ce classifieur s'affranchit de la condition d'indépendance des attributs. Nous avons observé le phénomène inverse. Le classifieur naïf de Bayes réagit bien mieux à la combinaison des critères et tire mieux parti de leur diversité en utilisant, par exemple, des critères considérant les étiquettes morphosyntaxiques. Les résultats obtenus sont résumés dans le tableau 7.34. Ce tableau permet de comparer les résultats avec ceux du critère jugé le plus fiable : *[2gr]-[lemme]-[differencie]-[mot]*. Les combinaisons de critères retenues sont détaillées ci-dessous.

Pour les noms :

- *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de ±4 mots ;
- *[2gr]-[ems]-[non-ordonne]-[mot]* avec un contexte de ±1 mot ;
- *[1gr]-[jeton]-[differencie]-[mot-plein]* avec un contexte de ±3 mots pleins ;
- *[1gr]-[lemme]-[non-ordonne]-[mot]* avec un contexte de ±2 mots.

Pour les adjectifs :

- *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de ±3 mots ;
- *[2gr]-[jeton]-[non-ordonne]-[mot-plein]* avec un contexte de ±2 mots ;
- *[2gr]-[ems]-[differencie]-[mot-plein]* avec un contexte de ±1 mot.

Pour les verbes :

- *[2gr]-[lemme]-[differencie]-[mot]* avec un contexte de ±4 mots décalé de un mot vers la droite ;
- *[1gr]-[ems]-[ordonne]-[mot]* avec un contexte de ±2 mots décalé de un mot vers la droite ;
- *[1gr]-[lemme]-[non-ordonne]-[mot-plein]* avec un contexte de ±1 mot décalé de un mot vers la droite ;
- *[2gr]-[jeton]-[ordonne]-[mot]* avec un contexte de ±4 mots décalé de un mot vers la droite.

L'amélioration obtenue est substantielle mais elle n'est pas phénoménale. Pour le classifieur `TNB(0,00)`, l'amélioration réalisée est de 2,9% pour les noms, 1,0% pour les adjectifs et 3,9% pour les verbes. Pour le classifieur `NB(0,98)` la combinaison des critères permet d'augmenter le rappel au détriment de la précision. L'augmentation du rappel est cependant bien supérieure à la dégradation de la précision, la performance globale est ainsi améliorée.

	Critère évalué indépendamment			Combinaison de critères		
	Noms	Adjectifs	Verbes	Noms	Adjectifs	Verbes
Précision `TNB(0,00)`	83,6%	77,9%	74,0%	86,5%	79,0%	77,9%
Précision `NB(0,98)`	92,8%	91,8%	88,1%	90,7%	87,9%	83,0%
Rappel `NB(0,98)`	69,3%	58,7%	59,9%	82,2%	68,8%	74,0%
Performance `NB(0,98)`	0,76	0,69	0,69	0,80	0,73	0,73

Tableau 7.34 – Comparaison des performances réalisées en combinant et sans combiner des critères.

		Noms	Adjectifs	Verbes
Combinaison de critères	Précision ou rappel	86,5%	79%	77,9%
	Entropie	1,9	2,3	3,1
Meilleur classifieur de SENSEVAL-1	Précision	83,3%	76,6%	70,5%
	Rappel	83,3%	73,8%	69,7%
	Entropie	1,9	1,7	1,9

Tableau 7.35 – Comparaison des performances que nous obtenons avec celles du meilleur classifieur de la campagne d'évaluation SENSEVAL-1. Ce tableau précise pour chaque catégorie de vocable la précision, le rappel et l'entropie de la répartition des occurrences sur les lexies.

Pouvons-nous comparer nos résultats avec ceux d'autres équipes ? La comparaison est difficile étant donné que nous ne travaillons ni sur la même langue, ni sur le même corpus, ni sur les mêmes vocables et que le dictionnaire utilisé n'est pas le même (cf. section 7.2.3 page 161). Le tableau 7.35 donne cependant, **à titre indicatif**, la comparaison de la performance du classifieur `TNB(0,00)` en combinant les critères, avec celle du meilleur classifieur de la campagne d'évaluation SENSEVAL-1 (cf. tableau 7.1 page 162). Nous obtenons systématiquement une précision et un rappel supérieur avec pourtant une entropie de la répartition des occurrences sur les lexies plus importante laissant supposer que notre tâche était plus difficile.

Dans la section 8.3 nous discutons de pistes pouvant amener une amélioration plus substantielle en utilisant d'autres types de critères.

Nous tenons ici à modérer les résultats que nous obtenons en combinant les critères. En effet, la technique utilisée dans la section 7.5 pour déterminer une bonne combinaison de critères s'apparente à de l'apprentissage supervisé manuel. L'apprentissage consiste ici à sélectionner le critère qui produit la plus grande augmentation de la précision lorsqu'il est utilisé conjointement avec le critère, puis la combinaison de critères de départ. Cependant, l'apprentissage manuel et l'évaluation se font sur l'ensemble du corpus sans utiliser une technique, comme la validation croisée k-fois, permettant d'estimer l'erreur réelle. Notre approche pour sélectionner la meilleure combinaison de critères est donc exposée à un risque de surapprentissage puisqu'elle n'est basée que sur une mesure de l'erreur apparente (cf. section 6.2.4 et 6.2.5). En fait, bien que dans une moindre mesure, les choix successifs réalisés au cours des sections 7.3 et 7.4 concernant les critères basés sur les unigrammes, les bigrammes et les trigrammes, sont exposés aux mêmes risques.

7.7 Fiabilité des résultats présentés

7.7.1 Comment limiter le risque d'erreur ?

Dans ce chapitre, qui constitue l'objet principal de cette thèse, nous avons effectué un très grand nombre d'expériences. Par exemple, la simple réalisation du tableau 7.4 a nécessité l'application de 11 520 critères sur notre corpus. Ces 11 520 critères ont généré 11 520 fichiers de données d'apprentissage sur lesquels il a fallu réaliser $11\,520 \times 10 = 110\,520$ apprentissages et 110 520 évaluations. Pour donner un autre ordre de grandeur, l'ensemble des données expérimentales utilisées (corpus, critères, scripts, etc.) et générées (données d'apprentissage, évaluations, etc.) pour le chapitre 7 occupent sur le disque une place de 16,8 Go et se répartissent en plus de 385 000 fichiers répartis en plus de 12 600 répertoires. Il est donc très difficile de garantir qu'aucune erreur ne se soit glissée dans notre travail. Pour limiter au maximum le risque d'erreur, nous avons adopté une attitude rigoureuse à tous les niveaux.

Cette rigueur commence au niveau de la constitution du corpus avec, par exemple, la vérification et la restauration de l'intégrité de la lemmatisation des occurrences des vocables de notre étude (cf. section 4.4).

Dans la section 5.2, nous exposons la méthodologie adoptée pour notre étude des critères de désambiguïsation lexicale. Nous validons cette méthodologie par une étude préliminaire, section 5.3. Cette étude préliminaire nous sert également d'amorce pour évaluer et adapter des algorithmes de classification pour notre étude. Nous dédions un chapitre complet (chapitre 6) à ce problème de classification supervisée car nous considérons qu'il s'agit d'un élément important et sensible dans notre approche.

Nous commençons le chapitre 7 par une section où nous positionnons notre étude par rapport aux études comparables menées par d'autres équipes et où nous décrivons notre protocole expérimental. Comme nous l'expliquons dans la section 7.2.4 concernant ce protocole, nous utilisons une méthode de validation croisée k-fois pour estimer l'erreur réelle. La précision de la désambiguïsation calculée en se basant sur une estimation de l'erreur réelle est bien moins bonne mais beaucoup plus réaliste que si elle était calculée en se basant sur une mesure de l'erreur apparente [6].

7.7.2 Rigueur dans la conception des applications

Les applications développées, et notamment (Win/DOS)LoX et AlgoWSD constituent un point sensible de ce travail de thèse. En effet, (Win/DOS)LoX est utilisée à tous les niveaux (constitution du corpus et application des critères) et (Win/DOS)LoX et AlgoWSD sont utilisées de manière intensive pour tous les traitements qui concernent ce chapitre. La réalisation de ces applications a donc retenu toute notre attention. Notre objectif lors de leur réalisation a été d'aboutir à un code clair, commenté, tirant parti de la puissance de la programmation objet et des mécanismes d'abstraction, mettant en œuvre une politique de gestion des exceptions et de génération de comptes rendus du déroulement des traitements.

La programmation objet et les mécanismes d'abstraction permettent d'éviter au maximum la redondance du code et apportent une meilleure clarté et une meilleure

6. À titre de comparaison, en utilisant le critère *[2gr]-[lemme]-[differencie]-[mot]*, la précision de la désambiguïsation moyenne pour les 60 vocables est de 74,6% avec le classifieur `TPCM(0,00)` et de 76,5% avec le classifieur `TNB(0,00)`. En mesurant cette précision d'après l'erreur apparente, c'est-à-dire en réalisant un apprentissage sur l'ensemble du corpus puis une évaluation sur ce même corpus, pour le même critère et les mêmes classifieurs, la précision serait respectivement de 91,4% et 99,9% !

lisibilité. D'autre part, cette non-redondance augmente grandement l'utilisation de chacune des lignes du code, ce qui permet de détecter d'éventuelles erreurs beaucoup plus rapidement. La réalisation de la bibliothèque LoX poursuit exactement les mêmes objectifs.

Chaque application génère un compte rendu du déroulement du traitement. Ce compte rendu permet de savoir si le traitement s'est bien effectué ou pas. En cas d'erreur, une information permettant de comprendre l'erreur est fournie. En aucun cas il ne faut qu'une erreur passe inaperçue, aussi mettons nous en œuvre une politique de gestion des exceptions. Cette politique permet, entre autre, d'alimenter le compte rendu.

7.7.3 Rigueur dans l'expérimentation

Nous avons tenté de toujours garder un regard critique sur les résultats de nos expériences : les résultats sont-ils attendus, sont-ils cohérents, sont-ils surprenants ? En cas de doute, nous analysions en détail, et étape par étape, l'ensemble de l'expérience pour vérifier si une erreur n'était pas survenue. Lorsque c'était possible, nous avons vérifié nos résultats en effectuant des recoupements. Par exemple, la précision obtenue en utilisant un critère basé sur les bigrammes de la forme *[2gr]-[lemme]-[ordonne]-[<param3>]* avec un contexte de plus ou moins un mot doit être la même que celle obtenue en utilisant le critère correspondant basé sur les unigrammes avec un contexte de plus ou moins un mot.

Dans un souci de non-redondance, nous avons choisi d'utiliser des règles génériques, et donc paramétrables, permettant de modéliser toute une famille de règles (cf. section 7.3.4).

Nous avons également choisi d'utiliser une cascade de scripts, pensée au préalable de manière à être largement réutilisable, pour automatiser chaque expérience. Cette façon de procéder a été retenue pour de multiples raisons.

- Cette cascade de scripts est réutilisable pour chacune des expériences moyennant des adaptations mineures.
- Elle permet d'éviter au maximum les interventions manuelles, sources inévitables d'erreurs.
- Elle permet de relancer très facilement une expérience après la détection d'une erreur ou pour vérifications ultérieures sans avoir à se remémorer chacune des étapes à effectuer.
- Elle permet enfin d'éviter la redondance des appels des traitements. Ainsi, dans le cas où une erreur est présente, elle ne concerne pas un traitement isolé mais toute une batterie de traitements, ce qui augmente la visibilité de l'erreur. La contrepartie étant que si une erreur passe inaperçue, elle a de forte chance d'être répercutée sur un grand nombre d'expériences. D'un autre côté, lorsqu'un traitement est effectué sans erreur, tous les traitements de la même famille ont de fortes chances de l'être aussi.

La synthèse et le regroupement des résultats sont également effectués par des scripts et des petites applications dédiées écrites en `C++`.

De plus, nous effectuons une vérification manuelle d'un grand nombre de données d'apprentissage pour vérifier la bonne application des critères. Cette opération est fastidieuse et ne peut être effectuée sur l'ensemble des données générées. Nous effectuons cette opération uniquement pour un ou deux vocables par catégorie grammaticale, sur une ou deux occurrences de chacun de ces vocables et pour une ou deux tailles de fenêtre de chaque famille de critères. L'entière automatisation des traitements est le gage que

si aucune erreur n'est détectée sur l'échantillon des fichiers vérifiés, il est peu probable qu'une erreur se glisse parmi les autres fichiers.

Enfin, nous conservons toutes les données utilisées et produites pour une expérience donnée (comptes rendus de chacun des traitements compris). Cet archivage nous permet de pouvoir contrôler *a posteriori* de manière complète chacune de nos expériences et éventuellement, lorsqu'une erreur est détectée, de pouvoir retracer ses origines et recommencer l'expérience à moindre coût.

Alors que nous étions à la fin des expériences de ce chapitre, nous avons constaté une erreur dans l'écriture d'un critère incriminant la quasi totalité des expériences de la section 7.3 (« Critères basés sur les cooccurrences »). Une première analyse nous a permis de constater que cette erreur était mineure et avait un impact sur les résultats (les précisions d'étiquetage) de l'ordre de 0,01% et toujours inférieur à 0,2%. La rigueur apportée au protocole expérimental, le souci apporté à l'automatisation des traitements et à l'archivage des résultats nous a permis de détecter cette erreur, d'en trouver les origines et de recommencer l'intégralité des expériences incriminées en l'espace de quelques jours (notamment en raison de la durée des traitements).

Chapitre 8

Conclusions

8.1 Rappel des objectifs

Il existe un grand nombre d'approches pour aborder le problème de la désambiguïsation lexicale automatique. Celles qui sont basées sur des corpus lexicalement désambiguïsés sont celles qui obtiennent actuellement les meilleurs résultats. Ce type d'approche, qui cherche à résoudre le problème en utilisant des techniques de classification supervisée, est adopté dans de nombreuses études récentes, comme l'attestent les successives campagnes d'évaluation SENSEVAL.

Notre travail se situait bien dans ce cadre. Cependant, notre objectif n'était pas de produire une nouvelle méthode de désambiguïsation lexicale automatique prête à l'emploi. Notre objectif était de proposer une étude rigoureuse, systématique et approfondie des critères pour la désambiguïsation lexicale automatique supervisée pour le français.

Pour mener à bien cette étude, nous avions besoin d'un corpus lexicalement désambiguïsé suivant l'inventaire des lexies d'un dictionnaire donné, pour permettre l'apprentissage et l'évaluation d'algorithmes de classification supervisée. Dès lors, nous étions confronté à deux difficultés majeures, récurrentes en désambiguïsation lexicale automatique :

- la première difficulté réside dans l'inadéquation des dictionnaires traditionnels pour cette tâche ;
- la seconde provient du manque de corpus lexicalement désambiguïsés sur lesquels des méthodes d'apprentissage supervisé peuvent être entraînées ; ce manque se transformant même en absence totale pour une langue comme le français.

Pour ces multiples raisons, notre équipe a entrepris la construction d'un dictionnaire distributionnel en se basant sur un ensemble de critères différentiels stricts. Ce dictionnaire comporte pour l'instant la description détaillée de 20 noms, 20 adjectifs et 20 verbes, et a été utilisé pour étiqueter manuellement chacune des 53 796 occurrences de ces 60 vocables dans le corpus du projet SYNTSEM (corpus composé de textes de genres variés comportant 6 468 522 *mots*).

La constitution du corpus, son étiquetage lexical manuel, ainsi que la modélisation des critères pour la désambiguïsation lexicale nous ont amené à concevoir nos propres outils. Tous ces outils ayant en commun un langage de requête, nous avons tout d'abord développé une bibliothèque `C++`, que nous avons ensuite utilisée pour réaliser les applications dont nous avions besoin.

Tous les éléments étaient alors en place pour débuter notre étude des critères pour la désambiguïsation lexicale.

8.2 Bilan

Les ressources dont nous disposions nous ont permis d'étudier un grand nombre de critères basés sur la cooccurrence de mots, et plus généralement de n-grammes (juxtaposition de un ou plusieurs mots), dans le contexte de 60 mots polysémiques cibles. Les critères avec lesquels nous avons obtenu les meilleurs résultats considèrent la forme lemmatisée de tous les bigrammes contenus dans un contexte de quatre mots à gauche et à droite pour les noms, un contexte de trois mots à gauche et à droite pour les adjectifs, et un contexte dissymétrique de trois mots à gauche et de cinq mots à droite pour les verbes. Ces critères distinguent les bigrammes suivant qu'ils se trouvent à gauche, à droite ou qu'ils contiennent le mot à désambiguïser. Ils permettent d'obtenir une précision de désambiguïsation de toutes les instances (*i.e.* précision égale au rappel) de 83,6% pour les noms, 77,9% pour les adjectifs et 74,35% pour les verbes. Lorsque nous ne cherchons pas à étiqueter toutes les instances, nous obtenons une précision (respectivement un rappel) de 92.8% (respectivement de 69.3%) pour les noms, de 91.8% (respectivement de 58.7%) pour les adjectifs et de 88.1% (respectivement de 60%) pour les verbes. Ces résultats sont d'autant plus encourageants que le découpage des entrées, au sein de notre dictionnaire, est relativement fin, puisque les noms, les adjectifs et les verbes comportent en moyenne respectivement 14,2, 14,1 et 47,4 lexies par vocable, l'entropie de la répartition des occurrences des lexies dans notre corpus (le corpus du projet SyntSem) étant de 1,9 pour les noms, 2,3 pour les adjectifs et 3,1 pour les verbes.

En dehors de ces considérations purement quantitatives, notre principale contribution comporte deux facettes.

La première facette réside dans le développement d'une bibliothèque `C++` qui implémente un langage élaboré et expressif d'interrogation de corpus, basé sur des *méta-expressions régulières*. Cette bibliothèque nous a permis de développer deux applications :

- un puissant outil de recherche et d'observation de phénomènes linguistiques dans les corpus écrits qui se décline en deux versions, la première possédant une interface graphique et la seconde étant orientée *ligne de commande* ;
- un concordancier/étiqueteur puissant.

La seconde facette de notre contribution réside dans notre étude systématique, rigoureuse et approfondie des critères pour la désambiguïsation lexicale. Il s'agit probablement de la première étude de cette ampleur réalisée dans un cadre unifié. Cette étude nous a permis de confirmer certains résultats de la littérature, comme :

- l'importance des petits contextes et de l'ordre des mots;
- l'importance du nom ou de l'adverbe adjacent pour désambiguïser un adjectif ;
- l'importance d'un adjectif adjacent, ou d'un nom dans un micro-contexte, pour désambiguïser un nom ;
- l'importance du nom dans la zone post-verbale pour désambiguïser un verbe, incitant, par là-même, à l'emploi de contextes dissymétriques pour cette catégorie grammaticale.

Nous avons également obtenu des résultats plus originaux allant parfois à l'encontre de certaines pratiques dans le domaine, comme :

- l'importance des mots grammaticaux dont le retrait dégrade pratiquement systématiquement les performances ;
- l'influence positive de la variabilité des indices sur la précision de la désambiguïsation ;

- le fait que les bigrammes considérés seuls donnent de meilleurs résultats que les unigrammes considérés seuls ;
- l'impact modeste de la lemmatisation pour les unigrammes ;
- le fait que les n-grammes ne doivent pas forcément contenir ou jouxter le mot à désambiguïser.

8.3 Perspectives

Nous ne parlerons pas des perspectives d'amélioration que nous prévoyons pour la bibliothèque LoX ainsi que pour les applications (Win/DOS)LoX et CooLoX puisque nous les avons déjà abordées dans les sections 3.4.14 et 3.5.3.

Les perspectives d'amélioration de la précision de la désambiguïsation sont nombreuses. Elles sont attendues aussi bien de la part des algorithmes de classification, du dictionnaire distributionnel, que des critères de désambiguïsation lexicale qu'il reste à étudier. Nous consacrons les trois sections qui suivent à chacun de ces trois aspects. Il est clair qu'à terme, pour parvenir à une précision de désambiguïsation lexicale maximale, l'affinage des algorithmes de classification et des critères de désambiguïsation devra être fait individuellement pour chaque vocable, et non simplement au niveau des catégories grammaticales.

Dans les deux dernières sections, nous évoquons d'autres perspectives qui ne concernent pas directement l'amélioration de la précision de la désambiguïsation. Il s'agit de la constitution interactive du dictionnaire et de l'étiquetage incrémental de corpus.

Algorithmes de classification

Dans cette étude, nous avons utilisé principalement deux types de classifieurs : le premier est basé sur une liste de décisions (`PCM(0,00)` et `PCM(0,57)`), le second correspond au classifieur naïf de Bayes (`NB(0,00)` et `NB(0,98)`). Nous avons parfois fait intervenir un troisième classifieur entrant dans la classe des algorithmes d'apprentissage basé sur les instances : le classifieur PEBLS. Nous avons observé que ces classifieurs réagissent de manière différente, voire parfois contradictoire, aux critères étudiés. Il serait intéressant de faire intervenir d'autres méthodes de classification comme des méthodes basées sur des arbres de décision ou sur de la programmation logique inductive [1]. Le recours à plusieurs techniques de classification répond au triple besoin qui suit.

1. Notre objectif est d'améliorer la précision de la désambiguïsation lexicale. Dans notre approche, l'algorithme de classification utilisé a un impact important sur cette précision. Or, il n'existe pas vraiment de classifieur qui soit meilleur qu'un autre dans l'absolu. L'évaluation de plusieurs classifieurs permet de sélectionner celui qui donne les meilleurs résultats.
2. D'autre part, il y a une interaction non négligeable entre les critères et les classifieurs. Il est donc important d'utiliser plusieurs techniques de classification pour porter un jugement sur un critère donné ou pour comparer deux critères entre eux.
3. Il est enfin tout à fait possible que la réalisation d'un classifieur tirant parti des décisions prises par plusieurs classifieurs puisse permettre d'aboutir à une performance (en terme de précision globale ou en terme de compromis *gain/rappel*)

1. Concernant la programmation logique inductive, voir, par exemple, le projet Aleph (`http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.html`).

supérieure à la performance individuelle du meilleur des classifieurs utilisés (Zavrel *et al.*, 2000).

Bien que les avancées futures, en désambiguïsation lexicale automatique supervisée, soient majoritairement à attendre de la part de l'amélioration des critères et des sources d'information utilisées (Pedersen, 2001), les algorithmes d'apprentissage supervisé ont aussi leur mot à dire puisque ce sont eux qui permettent de tirer parti de l'information retournée par les critères. Des efforts doivent donc parallèlement ou conjointement être menés au niveau des algorithmes d'apprentissage. À ce sujet, il faut noter que le projet WEKA (Witten & Frank, 2000) met à notre disposition une grande palette d'algorithmes de classification qui commencent à être utilisés dans notre domaine.

D'un autre côté, des travaux commencent à combiner la recherche des critères les plus pertinents avec la recherche des paramètres optimaux du classifieur utilisé. Ainsi, dans une étude récente, Daelemans, Hoste, Meulder et Naudts (2003) montrent comment obtenir une amélioration significative des performances en réalisant une optimisation simultanée des paramètres de l'algorithme d'apprentissage et de la sélection des indices. En raison de l'explosion combinatoire qu'engendre ce type d'optimisation, Daelemans *et al.* la réalise en utilisant un algorithme génétique.

Dictionnaire distributionnel

La réalisation du dictionnaire distributionnel n'est absolument pas du ressort de ce travail de thèse et constitue d'ailleurs l'objet d'un autre travail de thèse en cours (celui de Delphine Reymond). L'étude présentée ici est tout de même tributaire de ce dictionnaire et des perspectives d'amélioration sont à attendre de modifications ou corrections qui lui seront apportées. Ce dictionnaire ne dresse pas la liste des sens d'un mot mais plutôt une liste d'usages, chaque usage identifié correspondant à une lexie. Les expressions figées et semi-figées se voient chacune attribuer une lexie. C'est pourquoi certains vocables comportent un grand nombre de lexies. Ce dictionnaire est une ébauche et certaines entrées posent des problèmes (des lexies doivent par exemple être fusionnées). Encore très jeune, il ne comporte actuellement que les 60 vocables de notre étude et est susceptible d'évoluer fortement.

Actuellement, l'un de ses défauts est qu'il n'est pas consultable d'un point de vue informatique. En effet, son format est très proche d'un format papier puisqu'il est disponible au format *.doc* (*Microsoft Word*) ou *.pdf* (*Adobe Acrobat Reader*). Une amélioration importante à lui apporter serait de le rendre consultable par un programme en le portant au format *xml* par exemple. Les expressions figées pourraient alors faire l'objet d'un traitement spécifique préalable à la mise en œuvre des techniques de classification automatique. En effet, les expressions figées sont facilement identifiables et sont souvent d'une fréquence trop faible (inférieure à trois ou quatre occurrences) pour faire l'objet d'un apprentissage. L'entrée du dictionnaire *pied* section C.5 constitue un bon exemple pouvant tirer parti d'un tel prétraitement des expressions figées. Il faut toutefois modérer cette perspective d'amélioration par le fait que le nombre d'occurrences des lexies comportant moins de six occurrences dans le corpus n'est que de 1176, ce qui représente seulement 2,2% des occurrences des 60 vocables du corpus, sachant en outre que ces 1176 occurrences ne correspondent pas toutes à des expressions figées.

Critères de désambiguïsation lexicale

Dans le chapitre 7 nous avons présenté une étude des critères pour la désambiguïsation lexicale que nous pouvions étudier avec les ressources dont nous disposions alors.

Nous avons donc étudié des critères basés sur des cooccurrences ou des n-grammes. Cette étude doit encore être complétée par l'étude d'autres sources d'information utiles pour la désambiguïsation comme :

1. des critères basés sur des relations syntaxiques binaires ;
2. l'utilisation de thésaurus ou d'autres sources d'information pour effectuer des généralisations sur les mots du contexte du mot à désambiguïser ;
3. des informations sur le thème du texte ;
4. l'utilisation des restrictions de sélection.

La source d'information correspondant à l'item 1 devrait être étudiée à court terme. En effet, dans le cadre du projet SYNTSEM, un marquage syntaxique peu profond faisant apparaître les constituants majeurs est en cours de réalisation. Ce marquage devrait nous permettre, ou du moins nous faciliter, la recherche de ces relations syntaxiques binaires. L'utilisation de ce type de relation ne devrait pas apporter d'amélioration importante pour les adjectifs. En effet, les noms en relation syntaxique avec un adjectif jouxtent dans la majorité des cas cet adjectif. Dans les autres cas, ils sont en général très proches. La source d'information correspondant à l'item 1 est donc probablement dans une grande mesure capturée par les critères que nous avons déjà étudiés. En revanche, il en va tout autrement pour les verbes. En effet, le nom sujet du verbe à désambiguïser peut prendre une grande variété de positions et être noyé dans le contexte. De plus, le sujet peut très bien être suffisamment éloigné du verbe pour ne pas se retrouver dans le contexte des quelques mots considérés par les critères que nous avons étudiés. La relation syntaxique binaire *sujet–verbe* peut donc cibler une information qui est soit diffuse dans les indices retournés par les critères que nous avons étudiés, soit absente de ces indices car située à une distance importante du mot à désambiguïser. Ce type de relation syntaxique peut donc apporter une amélioration de la précision de la désambiguïsation des verbes, et de manière symétrique, des noms.

La seconde source d'information (item 2) est plus délicate car elle nécessite l'existence d'un thésaurus informatisé pour le français dont nous ne disposons pas. Néanmoins, nous sommes en relation avec J.-L. Manguin du CRISCO (CNRS de Caen) qui nous fournit un dictionnaire de synonymes (Francois, Victorri & Manguin, 1999) généré de manière automatique en se basant sur des calculs de cliques. Nous comptons explorer cette piste pour mesurer l'amélioration de la précision de la désambiguïsation qu'elle peut apporter.

Nous pensons que l'information sur le thème (item 3) ne peut être directement induite par l'algorithme de classification en lui fournissant directement l'ensemble des indices générés par un contexte élargi. Cette information devrait plutôt faire l'objet d'un prétraitement dont l'objectif serait de déterminer automatiquement la thématique du texte du voisinage du mot à désambiguïser. Ce serait uniquement le résultat de ce prétraitement qui serait fourni en tant qu'indice à l'algorithme de classification. Des recherches (Ferret, Grau, Minel & Porhiel, 2001 ; Bigi & Smaïli, 2002 ; Brun, Smaïli & Haton, 2002 ; Ferret, 2002 ; etc.) sont en cours dans ce domaine, notamment dans le cadre de la classification automatique de document, de la recherche documentaire et des systèmes de résumé automatique. Nous n'avons pas encore mené d'investigations sur cette piste, mais considérons qu'il s'agit d'une piste incontournable que nous pensons étudier en nous appuyant largement sur des travaux déjà effectués (nous ne projetons, *a priori*, pas de réaliser nous-même un analyseur thématique).

L'information apportée par l'utilisation des restrictions de sélection (item 4) constitue certainement une information importante qui permettrait de capturer une partie des critères distributionnels stricts utilisés pour la réalisation du dictionnaire. Pour augmenter l'efficacité de ces restrictions, il faudrait également posséder une information

sur les traits syntaxico-sémantiques (humain, animal, végétal, inanimé concret, inanimé abstrait, locatif, temps, événement), voire sur les classes d'objets qui permettent une sous-catégorisation des traits (cf. Gross & Clas, 1997). L'utilisation des restrictions de sélection demandera certainement un effort important, mais il est probable qu'un tel effort soit justifié par l'amélioration de la précision que nous pouvons en attendre.

Constitution interactive du dictionnaire

Il existe actuellement un cloisonnement important entre ce travail de thèse et celui concernant la réalisation du dictionnaire distributionnel. Dans un premier temps, ce cloisonnement est souhaitable pour ne pas biaiser les résultats de notre étude. En effet, les entrées du dictionnaire distributionnel ne doivent pas être conçues avec pour objectif que les algorithmes de classification supervisée fonctionnent bien sur notre corpus. Pour la cohérence et la justesse de notre étude, il ne fallait pas que celle-ci interfère sur la réalisation du dictionnaire.

À terme, cette politique n'est certainement pas souhaitable. En effet, la réalisation du dictionnaire est une tâche fastidieuse, complexe et primordiale. Tous les moyens doivent être mis en œuvre pour faciliter et améliorer la réalisation d'un tel travail. De nombreuses techniques récentes peuvent intervenir dans la réalisation de cette tâche. Par exemple, l'algorithme *HyperLex* (Véronis, 2003) permet d'extraire automatiquement la liste des usages des mots d'un corpus. Un tel algorithme peut très bien être utilisé pour faciliter la recherche des usages (*i.e.* des lexies) d'un vocable. Cet algorithme peut également aider à savoir s'il faut subdiviser ou pas une lexie. D'un autre côté, les techniques utilisées dans cette étude peuvent permettre d'enrichir la description d'une lexie en faisant ressortir des critères de désambiguïsation fiables non encore identifiés lors de la réalisation des entrées du dictionnaire. En outre, lorsque les techniques mises en œuvre dans cette étude ne parviennent pas à identifier des critères de désambiguïsation fiables pour certaines lexies, il peut s'agir d'un indicateur laissant penser que ces lexies ne sont pas cohérentes et qu'il faut les fusionner ou les remanier.

Étiquetage incrémental de corpus

Des méthodes d'étiquetage incrémental de corpus ont été décrites section 2.5.5. L'objectif de telles méthodes est, par exemple, de réduire le coût de la constitution de grands corpus lexicalement désambiguïsés. La méthode classique consiste à étiqueter manuellement un petit ensemble d'exemples servant de données d'apprentissage à une méthode de désambiguïsation utilisée pour désambiguïser les exemples restants. Seuls les exemples dont le sens est automatiquement identifié avec une fiabilité suffisante sont effectivement désambiguïsés. Ces nouveaux exemples sont alors traités comme des exemples manuellement étiquetés pouvant à leur tour servir de données d'apprentissage. Les occurrences non désambiguïsées après plusieurs cycles restent à étiqueter manuellement. Cette technique peut permettre de réduire de manière importante le nombre d'occurrences à étiqueter à la main.

Pour ce type d'application, le classifieur doit chercher à maximiser la précision, même si cela se fait au détriment du rappel. La précision doit être très proche de la limite maximale déterminée par le degré d'accord entre plusieurs annotateurs humains (ITA, cf. section 6.3.1). Les classifieurs `TPCM(0,00)`, `PCM(0,00)`, `TNB(0,00)`, `NB(0,00)` ne sont pas du tout adaptés pour ce type d'application puisque leur objectif est l'étiquetage d'un maximum d'occurrences. Le classifieur `NB(0,98)` paraît déjà plus adapté puisqu'il a été défini pour maximiser la mesure de performance combinant le

gain et le rappel. Il devrait être possible d'obtenir des performances bien meilleures en cherchant à optimiser tous les paramètres spécifiquement pour ce problème. Il est, par exemple, possible de rechercher les critères les plus adaptés, non pas pour maximiser le rappel, mais plutôt pour maximiser la précision. Nous pouvons également affiner les paramètres de l'algorithme de classification pour maximiser la précision, ou combiner plusieurs classifieurs et ne prendre une décision qu'à l'unanimité des classifieurs. En fait, les moyens d'augmenter la précision au détriment du rappel sont nombreux et la précision déjà réalisée par le classifieur `NB(0,98)` est très encourageante pour une utilisation dans le cadre de l'étiquetage incrémental de corpus.

Annexe A

Manuel d'utilisation de WinLoX et DOSLoX

A.1 Présentation

A.1.1 Introduction

(Win/DOS)LoX [1] est un puissant outil de recherche et d'observation de phénomènes linguistiques dans les corpus écrits. Cet outil s'adresse donc aux lexicographes et aux chercheurs en linguistique.

Les linguistes habitués au travail sur les corpus écrits sont généralement familiers avec les expressions régulières. Ces expressions se situent au niveau des caractères et permettent de décrire de manière formelle des classes de chaînes de caractères. (Win/DOS)LoX utilise le formalisme des méta-expressions régulières de la bibliothèque LoX. Ces expressions se situent au niveau des mots et permettent de décrire de manière formelle des classes de suites de mots, donc des portions de texte.

(Win/DOS)LoX reconnaît deux types de corpus, les corpus de texte brut (format le plus classique et le plus simple), et les corpus étiquetés au format tabulaire (comme ceux générés par le logiciel Cordial Analyseur). Lorsqu'un corpus étiqueté est utilisé, il est possible de travailler directement sur les mots ou bien sur les étiquettes de ces mots (comme le lemme ou l'étiquette morphosyntaxique).

A.1.2 Possibilités

(Win/DOS)LoX permet de rechercher et de dénombrer des phénomènes linguistiques dans des corpus écrits. Un phénomène linguistique est décrit formellement par une méta expression régulière. Les portions du corpus décrites par la méta expression régulière sont appelées des correspondances (instance du phénomène linguistique). Pour chacune des correspondances l'application permet de générer diverses chaînes de caractères qui constituent le résultat du traitement. Un formalisme, appelé masque, permet de générer ces chaînes à partir des correspondances. Les masques possèdent un pouvoir

1. Nous utilisons l'acronyme (Win/DOS)LoX pour désigner indifféremment l'application WinLoX ou l'application DOSLoX. Site Internet : `http://laurent.audibert.free.fr/lox.htm` . Référencement : `http://www.biomath.jussieu.fr/ATALA/outil/doslox_winlox_audibert_-laurent.html` .

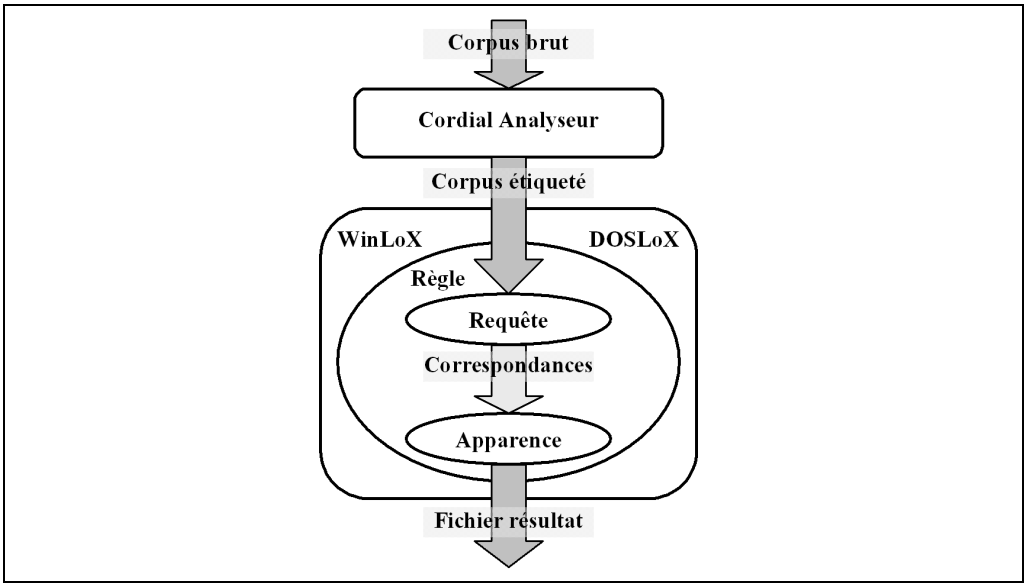


Figure A.1 – Représentation schématique de la chaîne du traitement de l'information dans laquelle (WIN/DOS)LOX prend place.

expressif assez grand pour générer des phrases (pour une utilisation du style concordancier), des références (pour retrouver la correspondance dans le corpus), ou d'autres chaînes pouvant servir à un traitement automatique ou manuel subséquent.

(WIN/DOS)LOX permet également d'extraire du corpus un sous-corpus particulier. Ce sous-corpus peut alors être d'une taille suffisamment raisonnable pour être traité manuellement et contenir de manière concentrée le phénomène étudié.

(WIN/DOS)LOX permet enfin de réaliser des étiquetages systématiques.

A.1.3 WINLOX et DOSLOX

WINLOX et DOSLOX ont exactement les mêmes fonctionnalités. La seule différence entre les deux est que DOSLOX est orienté *ligne de commande* tandis que WINLOX, qui ne fonctionne que sous *Microsoft Windows*, possède une interface graphique. DOSLOX et WINLOX sont entièrement compatibles dans le sens où ils travaillent sur les mêmes fichiers et peuvent donc les échanger.

DOSLOX peut être compilé indifféremment sur n'importe quelle plate-forme possédant un compilateur `C++` avec les bibliothèques standards auxquelles il faut adjoindre la bibliothèque REGEX++ (disponible sur le site *Boost.org* [2] .) pour les expressions régulières.

WINLOX dispose d'une interface graphique pour faciliter la saisie des paramètres. Il fonctionne sur *Microsoft Windows 98*, *Windows Millenium*, *Windows NT4.0* et *Windows 2000* et *XP*.

A.1.4 Fonctionnement général

La souplesse et la puissance expressive des requêtes et des masques permettent une diversité de traitement très importante. Nous allons décrire ici le schéma d'utilisation

2. Adresse Internet ; `http://www.boost.org/`

typique et simplifié des applications (WIN/DOS)LOX. La figure A.1 représente graphiquement la chaîne du traitement de l'information dans laquelle (WIN/DOS)LOX prend place.

Généralement, le corpus de travail n'est pas étiqueté. Or, pour tirer le meilleur parti des requêtes, il faut pouvoir faire référence au lemme des mots ou à leur étiquette morphosyntaxique. Une phase préalable consiste donc à étiqueter le corpus brut. Cet étiquetage peut se faire au moyen de diverses applications, comme CORDIAL ANALYSEUR par exemple.

Il faut ensuite écrire une requête qui modélise le phénomène linguistique à étudier. L'application de cette requête au corpus étiqueté produit des correspondances, une correspondance étant un objet interne qui désigne une portion de corpus décrite par la requête. Les correspondances passent enfin au travers d'un masque appelé apparence qui permet de produire le fichier résultat. Concrètement, un masque est écrit dans un langage de formatage qui précise comment la correspondance doit être interprétée pour produire un résultat.

Le fichier résultat ainsi produit peut, suivant les cas, être lu directement dans un éditeur de texte, être ouvert dans un tableur ou encore servir d'entrée à un programme effectuant un traitement subséquent.

A.1.5 Description des sections qui suivent

La section A.2 explique comment utiliser DOSLOX tandis que la section A.3 explique comment utiliser WINLOX.

Les sections A.4.1 à A.4.7 détaillent tous les paramètres de (WIN/DOS)LOX un par un. La description de chacun des paramètres comporte :

- une section WINLOX décrivant comment spécifier le, ou les, paramètres avec WINLOX ;
- une section DOSLOX décrivant la syntaxe du, ou des, paramètres dans le fichier paramètre ;
- une ou plusieurs sections explicitant la signification du, ou des, paramètres.

Dans la dernière section (A.5) nous traitons un exemple.

A.2 Utilisation de DOSLOX

A.2.1 Généralités

DOSLOX est directement exécuté à partir d'une *ligne de commande* (invite de commande DOS, Fenêtre DOS sous *Microsoft Windows*, shell sous *Unix/Linux* [3], etc.). Toutes les informations permettant de définir le traitement doivent se trouver dans un fichier texte.

A.2.2 Syntaxe de la *ligne de commande*

```
Invite_de_commande> DOSLoX <fichier>
```

<**fichier**>**:** chemin complet d'accès au fichier de paramètres.

3. À condition de compiler DOSLOX sous *Unix/Linux*.

A.2.3 Fichier de paramètres

Le fichier de paramètres est un simple fichier au format texte décrivant tous les paramètres du traitement à réaliser.

Toute ligne ne commençant pas par une chaîne permettant de préciser un paramètre est ignorée (ces lignes peuvent servir de commentaire par exemple).

Chaque ligne permettant de préciser un paramètre est de la forme :

`\<type_parametre>=<parametre>`.

Il ne peut y avoir deux paramètres sur une ligne et la définition d'un paramètre ne peut s'étaler sur plusieurs lignes.

Dans l'arborescence des répertoires, le fichier paramètre se trouve à un endroit donné. Toutes les références à des fichiers au sein du fichier paramètre doivent se faire de manière relative à l'emplacement de ce fichier paramètre. Il est ainsi possible de déplacer un projet sans que celui-ci soit altéré, à condition que les positions relatives entre tous les fichiers ne soit pas modifiées.

A.2.4 Liste des paramètres

Pour une explication détaillée sur chacun des paramètres il faut se référer à la section correspondante. Les paramètres permettant de définir le traitement sont :

`\act=<action>` voir section A.4.6.
`\opt_act=<opt>` voir section A.4.7.
`\regles=<chemin><<param>>...<<param>>` voir section A.4.4.
`\corpus=<chemin>` voir section A.4.1.
`\corpus_type=<type>` voir section A.4.2.
`\fic_prop=<chemin>` voir section A.4.3.
`\sep_prop=<nombre>` voir section A.4.3.
`\sep_mot=<nombre>` voir section A.4.3.
`\fic_res=<chemin>` voir section A.4.5.
`\fic_ref=<chemin>` voir section A.4.5.
`\fic_inf=<chemin>` voir section A.4.5.

A.3 Utilisation WinLoX

A.3.1 Généralités

L'interface graphique de WinLoX permet de spécifier tous les paramètres d'un traitement puis de réaliser ce traitement. Il est possible d'enregistrer ces paramètres dans un fichier. Le fichier ainsi généré est un fichier texte pouvant être repris pour des traitements ultérieurs indifféremment par DOSLoX ou par WinLoX.

Toutes les références à des fichiers (chemin) doivent se faire de manière absolue (c'est l'inverse pour DOSLoX). Au moment de l'enregistrement, les chemins sont convertis en chemins relatifs. Cela permet de pouvoir déplacer un projet sans avoir à spécifier de nouveau l'emplacement des fichiers (à condition que la position relative des fichiers ne soit pas modifiée). Lors du chargement d'un fichier paramètre, tous les chemins relatifs sont convertis en chemins absolus.

A.3.2 Menu *Fichier*

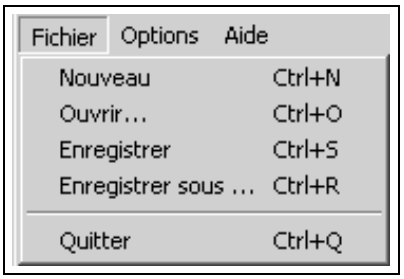


Figure A.2 – Menu *Fichier* de l'application WinLoX.

Ce menu, représenté figure A.2, permet essentiellement d'ouvrir (sous-menu *Ouvrir...*) et d'enregistrer (sous-menu *Enregistrer* et *Enregistrer sous...*) un fichier paramètre. Le sous-menu *Nouveau* permet de remettre tous les paramètres à leur valeur par défaut. Le sous-menu *Quitter* permet de quitter l'application.

A.3.3 Menu *Options*

Figure A.3 – Menu *Options* de l'application WinLoX.

Ce menu, représenté figure A.3, permet d'enregistrer l'application dans la base de registre. L'intérêt de cette opération est l'association des extensions avec les applications correspondantes, ainsi que la prise en compte des icônes. Cela est indispensable au bon fonctionnement de WinLoX. Cette opération n'est utile que lors de la première utilisation de WinLoX sur une machine donnée.

A.3.4 Menu *Aide*

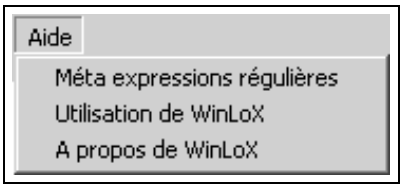


Figure A.4 – Menu *Aide* de l'application WinLoX.

Ce menu, représenté figure A.4, permet de visualiser :

- le fichier d'aide de WinLoX, dont le contenu est proche de cette annexe ;
- le fichier d'aide sur le formalisme des méta-expressions régulières et des masques, dont le contenu est proche de celui de la section 3.4 ;
- les informations du type version de WinLoX, auteurs, contacts, etc.

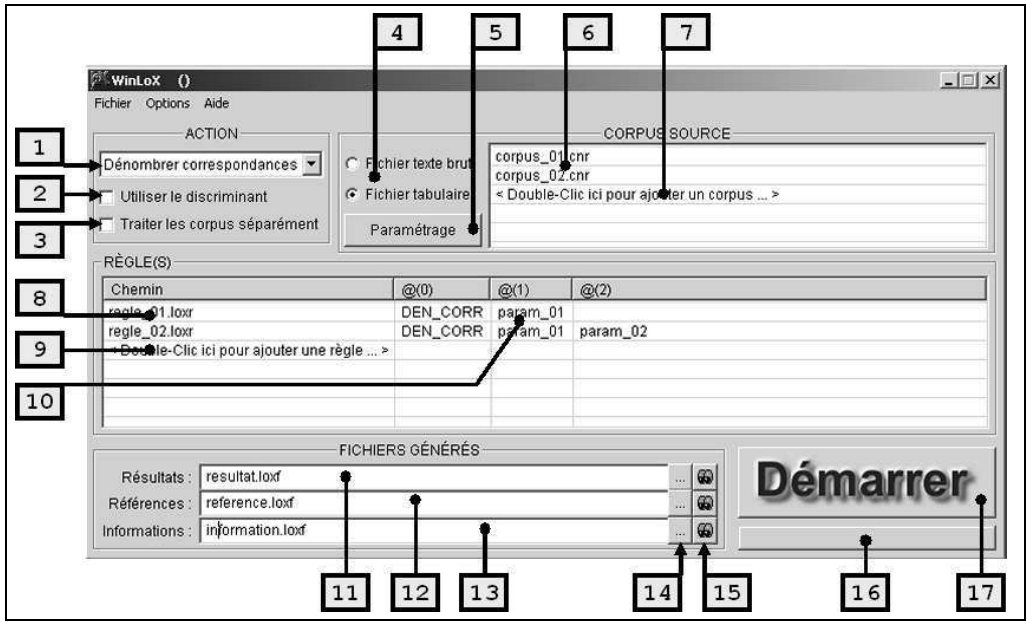


Figure A.5 – Fenêtre principale de l'application WINLOX.

A.3.5 Fenêtre principale

La fenêtre principale de l'application WINLOX est présentée figure A.5. Sur cette figure, les différentes zones qui permettent de spécifier les paramètres du traitement sont identifiées par des repères, l'explication succincte de ces paramètres est donnée ci-dessous. Pour plus d'information, il faut se référer à la section correspondante.

1. Cette liste déroulante permet de désigner le traitement à effectuer (cf. section A.4.6).
2. Si cette case est cochée, le masque discriminant des règles est utilisé. Cette option n'est pas disponible pour toutes les actions.
3. Si cette case est cochée, les corpus sources sont traités séparément. Autant de fichiers résultats et références que de corpus sources sont alors générés.
4. Permet de spécifier le format des corpus sources sur lesquels le traitement est réalisé (cf. section A.4.3).
5. Permet d'accéder au paramétrage des corpus tabulaires (cf. section A.3.8).
6. Un double-clic gauche de la souris sur un des corpus permet d'accéder à un explorateur qui permet de désigner un autre corpus. Un clic droit sur un corpus permet d'accéder au menu contextuel relatif aux corpus (cf. section A.3.7).
7. Un double-clic gauche sur cette ligne permet d'ajouter un nouveau corpus.
8. Un double-clic gauche de la souris sur une des règles permet d'accéder à la boîte de dialogue de prise en compte de cette règle (cf. section A.3.9). Il est alors, par exemple, possible de choisir une autre règle ou de modifier ses paramètres. Un clic droit sur une des règles permet d'accéder au menu contextuel relatif aux règles (cf. section A.3.6).
9. Un double-clic gauche sur cette ligne permet d'ajouter une nouvelle règle.
10. Dans cette zone sont affichés les paramètres des règles.

11. Affiche et permet de saisir le chemin du fichier résultat généré lors du traitement (cf. section A.4.5).
12. Affiche et permet de saisir le chemin du fichier référence généré lors du traitement (cf. section A.4.5).
13. Affiche et permet de saisir le chemin du fichier information généré lors du traitement (cf. section A.4.5).
14. Ce bouton permet d'invoquer un explorateur pour spécifier le chemin du fichier.
15. Ce bouton permet d'afficher le contenu du fichier.
16. Ce bouton permet de lancer le traitement.
17. Cette barre informe sur l'état d'avancement du traitement en cours.

A.3.6 Menu contextuel des règles

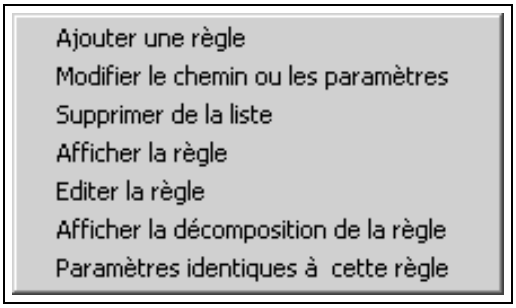


Figure A.6 – Menu flottant des règles de l'application WinLoX.

Ce menu, représenté figure A.6, s'obtient en cliquant sur le bouton droit dans la zone de liste des règles.

Ajouter une règle permet d'ajouter une nouvelle règle à la fin de la liste des règles. Cette action ouvre à la boîte de dialogue de prise en compte des règles (cf. section A.3.9).

Modifier le chemin ou les paramètres permet d'accéder à la boîte de dialogue de prise en compte des règles (cf. section A.3.9) pour modifier le chemin ou les paramètres de la règle sélectionnée.

Supprimer de la liste supprime la règle sélectionnée de la liste.

Afficher la règle permet de voir la règle sans pouvoir la modifier.

Éditer la règle ouvre la règle dans un éditeur de fichier permettant ainsi de la modifier.

Afficher la décomposition de la règle affiche la décomposition complète de la règle si celle-ci est syntaxiquement bien écrite. La décomposition permet parfois de comprendre les erreurs non syntaxiques que nous avons pu commettre en écrivant la règle.

Paramètres identiques à cette règle permet de recopier tous les paramètres de la règle sélectionnée sur les autres règles.

A.3.7 Menu contextuel des corpus

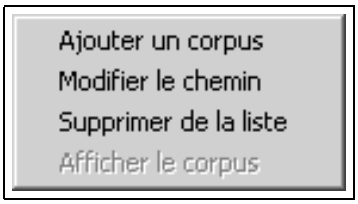


Figure A.7 – Menu flottant des corpus de l'application WINLOX.

Ce menu, représenté figure A.7, s'obtient en cliquant sur le bouton droit dans la zone de liste des corpus.

Ajouter un corpus permet d'ajouter un nouveau corpus à la fin de la liste des corpus.

Modifier le chemin permet de remplacer le corpus sélectionné dans la liste par un autre corpus.

Supprimer de la liste supprime le corpus sélectionné de la liste.

Afficher le corpus affiche le corpus sélectionné.

A.3.8 Paramétrage des fichiers tabulaires

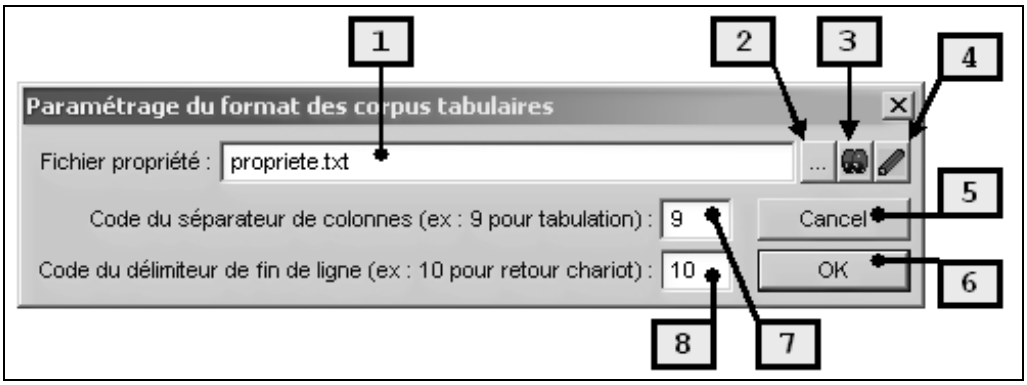


Figure A.8 – Paramétrage des fichiers tabulaires dans l'application WINLOX.

La fenêtre de paramétrage des fichiers tabulaires est présentée figure A.8. Sur cette figure, les différentes zones qui permettent de spécifier les paramètres des fichiers tabulaires sont identifiées par des repères, l'explication succincte de ces paramètre est donnée ci-dessous.

1. Cette zone d'édition permet de spécifier le chemin du fichier contenant la liste des propriétés qui identifient les colonnes du corpus (cf. section A.4.3).
2. Ce bouton permet d'invoquer un explorateur pour spécifier le chemin du fichier.
3. Ce bouton permet d'afficher le contenu du fichier.
4. Ce bouton permet d'éditer le contenu du fichier.
5. *Cancel* permet d'ignorer les données saisies et de fermer la boîte de dialogue.
6. *Ok* permet de valider les données saisies et de fermer la boîte de dialogue.
7. Cette zone d'édition permet de saisir le code qui correspond au caractère de séparateur de colonnes dans le corpus tabulaire.

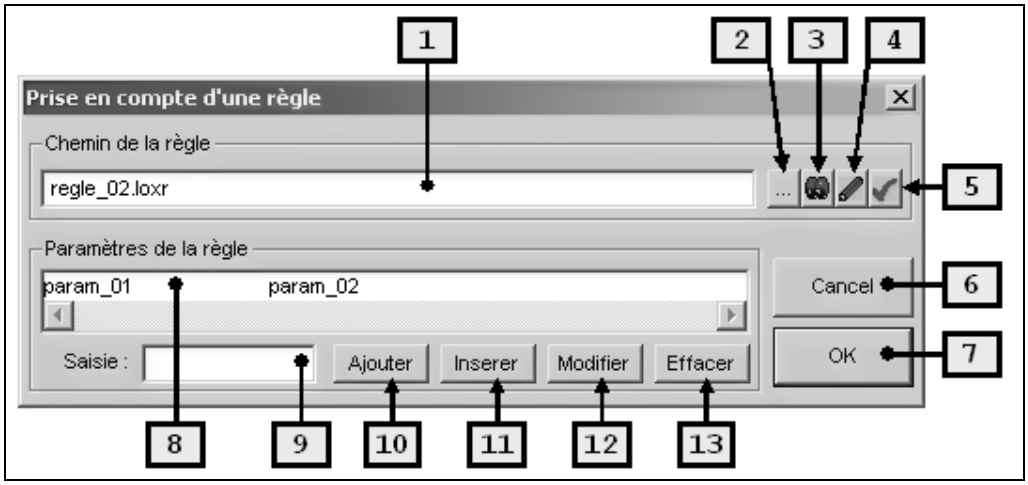


Figure A.9 – Prise en compte des règles dans l'application WinLoX.

8. Cette zone d'édition permet de saisir le code qui correspond au caractère qui délimite la fin de ligne dans le corpus tabulaire.

A.3.9 Prise en compte des règles

1. Cette zone d'édition permet de spécifier le chemin du fichier contenant la règle (cf. section A.4.4).
2. Ce bouton permet d'invoquer un explorateur pour spécifier le chemin du fichier.
3. Ce bouton permet d'afficher le contenu du fichier.
4. Ce bouton permet d'éditer le contenu du fichier.
5. Ce bouton permet de tester la validité de la règle et d'afficher sa décomposition.
6. *Cancel* permet d'ignorer les données saisies et de fermer la boîte de dialogue.
7. *Ok* permet de valider les données saisies et de fermer la boîte de dialogue.
8. Dans cette zone sont affichés les paramètres de la règle. Il est possible de sélectionner un paramètre pour préciser le paramètre qui est modifié, effacé ou encore la position du paramètre qui doit être inséré.
9. C'est la zone d'édition des paramètres.
10. Ce bouton permet d'ajouter le paramètre saisi à la fin de la liste des paramètres.
11. Ce bouton permet d'insérer le paramètre saisi devant le paramètre sélectionné.
12. Ce bouton permet de remplacer le paramètre sélectionné par le paramètre saisi.
13. Ce bouton permet d'effacer le paramètre sélectionné.

A.4 Paramètres

A.4.1 Corpus

Ce paramètre permet de spécifier les corpus sur lesquels le traitement est réalisé.

WinLoX

Un double-clic gauche de la souris sur un des corpus permet d'accéder à un explorateur qui permet de désigner un autre corpus. Un double-clic gauche sur la dernière

ligne (*Double-Clic ici pour ajouter un corpus...*) permet d'ajouter un nouveau corpus. Un clic droit sur un corpus permet d'accéder au menu contextuel relatif aux corpus représenté figure A.7.

DOSLoX

La syntaxe du paramètre est :

```
\corpus=<chemin>
```

où `<chemin>` est à remplacer par le chemin d'accès au fichier contenant le corpus. Cette commande doit être répétée autant de fois qu'il y a de corpus.

A.4.2 Type de corpus

WinLoX

Il suffit de cocher la case correspondant au bon type de corpus.

DOSLoX

La syntaxe du paramètre est :

```
\corpus_type=<type>
```

où `<type>` doit être remplacé par `texte` pour un fichier texte brut et par `tabulaire` pour un fichier tabulaire.

Description

Si le fichier contient un texte brut, la segmentation des mots se fait suivant le principe de la segmentation maximale. Dans une requête ou un masque, la référence à un mot se fait avec le mot clef *word*.

Dans un corpus tabulaire, chaque ligne correspond à un élément (*i.e.* un mot) tandis que chaque colonne correspond à une propriété de l'élément. La fin d'une ligne est identifiée par un délimiteur particulier. De même, un autre délimiteur particulier sépare les colonnes.

A.4.3 Format des Corpus tabulaires

WinLoX

Le paramétrage du format des corpus tabulaires se fait par l'intermédiaire de la boîte de dialogue représentée sur la figure A.8.

DOSLoX

La syntaxe du paramètre est :

```
\fic_prop=<fichier>
\sep_prop=<nombre_1>
\sep_mot=<nombre_2>
```

où `<fichier>` permet de spécifier le chemin du fichier contenant la liste des propriétés qui permet d'identifier les colonnes du corpus, `<nombre_1>` est le nombre correspondant au code du caractère qui sépare chaque propriété (*i.e.* qui sépare chaque colonne) dans le corpus et `<nombre_2>` est le nombre correspondant au code du caractère qui sépare chaque mot (*i.e.* qui délimite la fin de ligne) dans le corpus.

Description

Le fichier propriété est un fichier du même format que le corpus tabulaire. Ce fichier n'est utile que pour travailler avec des corpus tabulaires. Seule la première ligne du fichier est lue. Cette ligne est constituée d'un certain nombre de chaînes de caractères qui désignent le nom que l'utilisateur donne à chacune des colonnes du corpus. Ces noms sont séparés par le délimiteur de séparateur de propriété. Un nom de propriété ne doit pas contenir de caractère spécial (entre autres `"`, `(`, `)`, `[`, `]`) et ne doit pas correspondre à un nom réservé comme *index*, *exist*, *end*, *begin*, *word*, *next*, ou *prev*.

Dans le cadre de cette étude, le fichier propriété permettant d'identifier les différentes colonnes de notre corpus finalisé (cf. figure 4.14) est le suivant :

```
fichier → paragraphe → position → jeton → lemme →
ems → tgb → fonc → prop → smallems → lexie → freq ¶
```

Ce fichier ne contient qu'une ligne (bien que sa représentation ci-dessus en comporte deux), les séparateurs de colonnes sont des tabulations (représentées par le caractère → ci-dessus) et le séparateur de fin de ligne est un retour chariot (représenté par le caractère ¶ ci-dessus).

A.4.4 Règles

Ce paramètre permet de spécifier les règles et leurs paramètres associés.

WinLoX

Un double-clic gauche de la souris sur l'une des règles, ou sur la dernière ligne (*Double-Clic ici pour ajouter une règle...*) permet d'accéder à la boîte de dialogue de prise en compte des règles représentée sur la figure A.9. Un clic droit sur une règle permet d'accéder au menu contextuel relatif aux règles représenté figure A.6.

DOSLoX

La syntaxe du paramètre est :

```
\regles=<chemin><<param>><<param>>...<<param>>
```

où `<chemin>` est le chemin d'accès au fichier contenant la règle et `<param>` l'un des paramètres de la règle. Cette commande doit être répétée autant de fois qu'il y a de règles.

Description

Une règle se décompose en quatre parties. Le rôle exact de chacune de ces parties dépend du traitement effectué (*i.e.* dépend de l'action choisie). Toutes les parties ne sont pas obligatoirement utilisées pour chacun des traitements.

Dans le cas général, une règle est évaluée de la manière suivante :

- tout d'abord, la requête de la règle est évaluée et retourne un certain nombre de correspondances ;
- ensuite, pour chacune des correspondances retournées par la requête, les masques apparence, référence et discriminant sont générés.

Une règle est stockée dans un fichier texte dont un modèle est représenté ci-dessous:

```
// Commentaires
Requête :       <requete>
Apparence :     <masque>
Référence :     <masque>
Discriminant :  <masque>
```

L'utilisation classique de ces parties est explicitée ci-dessous.

- `Requête :` la requête permet de définir de façon formelle les portions de corpus auxquelles l'utilisateur s'intéresse (cf. section 3.4.11).
- `Apparence :` l'apparence est un masque (cf. section 3.4.12) qui décrit la chaîne de caractères qui doit être générée à partir de la correspondance. Plusieurs apparences peuvent être générées. Il peut donc y avoir plusieurs lignes commençant par « `Apparence :` ».
- `Référence :` la référence est un masque (cf. section 3.4.12) qui décrit la chaîne de caractères qui peut servir, par exemple, de référence pour retrouver la correspondance dans le corpus.
- `Discriminant :` le discriminant est un masque (cf. section 3.4.12) qui décrit la chaîne de caractères qui sert à discriminer les apparences identiques.

Chacune des quatre parties de la règle débute par un identifiant explicite (en gras dans le modèle de règle ci-dessus) et se termine soit par la fin du fichier, soit par un nouvel identifiant explicite. L'ordre des quatre parties n'a pas d'importance. Par souci d'intelligibilité des fichiers règles, il est possible d'écrire les expressions logiques ainsi que le masque sur plusieurs lignes (*i.e.* d'insérer des retours chariot). Toute ligne débutant par `//` est ignorée (lignes de commentaires).

Des paramètres peuvent être passés à une règle. Dans une règle, la référence à l'un de ses paramètres se fait en écrivant **`@(`**`<nombre>`**`)`**. Lors de l'utilisation d'une règle, la chaîne **`@(`**`<nombre>`**`)`** est remplacée par le `<nombre>`$^{\text{e}}$ paramètre passé à la règle. Le paramètre `@(0)` est singulier et désigne l'action en cours. Une utilisation pratique du paramètre `@(0)` est l'écriture de règles dont le comportement varie en fonction de l'action effectuée.

A.4.5 Fichiers

WinLoX

Il suffit de saisir le chemin absolu du fichier dans la zone de saisie. Pour simplifier cette opération, il est possible d'invoquer l'explorateur de *Microsoft Windows* en cliquant sur le bouton [...] (repère 14 sur la figure A.5).

DOSLoX

La syntaxe du paramètre est :

`\fic_res=``<chemin>`
`\fic_ref=``<chemin>`
`\fic_inf=``<chemin>`

où `<chemin>` est à remplacer par le chemin absolu d'accès aux fichiers résultat, référence et information.

Fichier résultat

Ce fichier est destiné à recevoir le résultat du traitement.

Fichier référence

C'est dans ce fichier que sont stockées les chaînes générées par le masque référence des règles. Les références sont stockées dans un tableau indicé par les apparences associées. L'utilisation la plus classique est de permettre de retrouver, dans le corpus, les correspondances ayant généré les apparences du fichier résultat.

Fichier information

Dans ce fichier se trouvent toutes les informations relatives au déroulement du traitement. Il faut consulter ce fichier pour connaître l'origine du problème lorsqu'une interruption (au sens informatique du terme) a interrompu le bon déroulement du traitement.

A.4.6 Actions

WinLoX

Il suffit de sélectionner l'action désirée dans le menu déroulant (repère 1 de la figure A.5) parmi :

- *Dénombrer correspondances* ;
- *Extraire correspondances* ;
- *Générer sous-corpus* ;
- *Étiquetage systématique.*

DOSLoX

La syntaxe du paramètre est la suivante :

```
\act=<action>
```

où `<action>` peut prendre les valeurs suivantes :

- `DEN_CORR` pour *Dénombrer correspondances* ;
- `EXT_CORR` pour *Extraire correspondances* ;
- `EXT_CORPUS` pour *Générer sous-corpus* ;
- `ETIQ_SYST` pour *Étiquetage systématique.*

Dénombrer des correspondances

Ce traitement permet de dénombrer les apparences issues des correspondances générées par l'application de la ou des règles sur le ou les corpus. Ces correspondances peuvent éventuellement être retrouvées dans le corpus grâce aux références générées par la ou les règles.

Dans le cas où le discriminant n'est pas utilisé, le fichier résultat est constitué, à l'issue du traitement, d'autant de lignes que de chaînes de caractères différentes générées par le masque apparence. Chacune des lignes contient la chaîne de caractères générée par le masque apparence suivi d'un séparateur de colonne puis du nombre de fois que cette chaîne a été générée.

Dans le cas où le discriminant n'est pas utilisé, le fichier référence comporte également autant de lignes que de chaînes de caractères différentes générées par le masque apparence. Chacune des lignes contient la chaîne de caractères générée par le masque apparence suivie de la liste des chaînes de caractères générées par le masque référence. Chacune de ces chaînes est séparée par un séparateur de colonne. Il peut très bien y avoir plusieurs références identiques sur une même ligne.

Dans le cas où le discriminant est utilisé, le fichier résultat contient, à l'issue du traitement, un tableau à deux dimensions. La première colonne contient la liste des chaînes de caractères différentes générées par le masque apparence. La première ligne contient la liste des chaînes de caractères différentes générées par le masque discriminant. Une case, à l'intersection de la colonne identifiée par la chaîne X (générée par le masque discriminant) et de la ligne identifiée par la chaîne Y (générée par le masque apparence), contient le nombre de fois que le couple $\{ X / Y \}$ a été généré.

Dans le cas où le discriminant est utilisé, le fichier référence comporte autant de lignes que de couples *{ chaîne générée par le masque apparence / chaîne générée par le masque discriminant }* différents générés. Chacune des lignes contient la chaîne de caractères générée par le masque apparence suivie de la chaîne de caractères générée par le masque discriminant suivie de la liste des chaînes de caractères générées par le masque référence. Chacune de ces chaînes étant séparée par un séparateur de colonne. Il peut très bien y avoir plusieurs références identiques sur une même ligne.

Options de l'action :

- *Utiliser le discriminant* permet de préciser s'il faut discriminer l'apparence d'une correspondance en fonction du discriminant de la règle ou pas.
- *Traiter les corpus séparément* permet de spécifier s'il faut que les corpus soient traités indépendamment les uns des autres ou pas.

Extraire des correspondances

Cette action écrit directement et séquentiellement les chaînes de caractères générées par le masque apparence dans le fichier résultat. Les masques référence et discriminant des règles sont ignorés pour cette action.

La seule option de l'action disponible est *traiter les corpus séparément*. Elle permet de spécifier s'il faut que les corpus soient traités indépendamment les uns des autres ou pas.

Générer des sous-corpus

Cette action extrait les portions de corpus désignées par les correspondances (correspondances générées par l'application de la ou des règles sur le ou les corpus) pour générer, dans le fichier résultat, un sous-corpus. Les masques apparence, référence et discriminant des règles sont ignorés pour cette action. Il peut arriver que des portions de corpus soient décrites par plusieurs correspondances (chevauchement des correspondances). Le sous-corpus généré ne contient aucun doublon. Dans le sous-corpus nous trouvons donc toutes les portions décrites par les correspondances et rien que les portions décrites par les correspondances, sans doublon et dans l'ordre où ces portions apparaissent dans le corpus source.

Cette action est en fait un cas particulier de l'action précédente. En effet, il est possible d'obtenir le même résultat avec une règle appropriée (mais plus complexe à écrire) en utilisant l'action *Extraire correspondances*.

La seule option de l'action disponible est *traiter les corpus séparément*. Elle permet de spécifier s'il faut que les corpus soient traités indépendamment les uns des autres ou pas.

Étiquetage systématique de corpus

Cette action permet de modifier une propriété des mots dans le corpus. Les mots à modifier sont décrits par la requête de la règle. Cette requête doit générer au maximum une correspondance à la fois qui ne décrit qu'un mot. La propriété à modifier est désignée par la chaîne générée par le masque discriminant de la règle. La nouvelle valeur de la propriété à modifier est désignée par l'unique masque apparence de la règle. Cette action ne modifie pas le corpus original, elle génère un nouveau corpus désigné dans le fichier résultat.

La seule option de l'action disponible est *traiter les corpus séparément*. Elle permet de spécifier s'il faut que les corpus soient traités indépendamment les uns des autres ou pas.

A.4.7 Options de l'action

Les options de l'action permettent de moduler le résultat du traitement. Il y a deux options distinctes : *Utiliser le discriminant* et *Traiter les corpus séparément*. Les effets de ces options sont décrits dans la section précédente.

WinLoX

Le dénombrement se fait sans utiliser le discriminant quand la case à cocher *Utiliser le discriminant* (repère 2 de la figure A.5) est décochée. Sinon il se fait en utilisant le discriminant.

Tous les corpus sont traités ensemble quand la case à cocher *Traiter les corpus séparément* (repère 3 de la figure A.5) est décochée. Si elle est cochée, les corpus sont traités séparément.

DOSLoX

La syntaxe du paramètre est :

```
\opt_act=<opt1>
\opt_act=<opt2>
```

Les valeurs possibles pour ces trois paramètres sont :

- `<opt1>=nodiscr` : pour ne pas utiliser le discriminant ;
- `<opt1>=discr` : pour utiliser le discriminant ;
- `<opt2>=corp_ens` : pour que les corpus soient traités ensemble ;
- `<opt2>=corp_sep` : pour que les corpus soient traités séparément.

A.5 Exemple de requête complexe

Pour illustrer le fonctionnement de l'application (Win|DOS)LoX et la capacité de représentation des méta-expressions régulières, nous allons effectuer un traitement complet répondant à une question complexe.

L'objectif est d'écrire une requête qui décrit les séquences contenant :

1. un verbe conjugué

```
\act=EXT_CORR
\opt_act=nodiscr
\opt_act=corp_ens
\corpus_type=tabulaire
\fic_prop=propriete.txt
\sep_prop=9
\sep_mot=10
\corpus=corpus.cnr
\regle=regle.loxr
\fic_res=res.loxf
\fic_ref=
\fic_inf=inf.loxf
```

Figure A.10 – Fichier `arguments.lox` correspondant aux arguments du traitement de l'exemple.

ou

2. un auxiliaire conjugué suivi de
3. zéro à quatre mots ne contenant pas de verbe suivi de
4. un verbe au participe passé

suivi de

5. zéro à quatre mots, autres que des verbes suivis de,
6. un verbe à l'infinitif.

La séquence 5-6 doit être répétée le maximum de fois possible et au minimum une fois. Le tout devant se trouver au sein d'une même phrase.

Une telle requête décrit par exemple les séquences suivantes :

- *commence à m'énerver* ;
- *pourrait commencer à vraiment m'énerver* ;
- *a vraiment pu se mettre, pour le moins, à s'énerver.*

Nous désirons générer un fichier contenant une séquence recherchée par ligne. L'action du traitement est donc *Extraire correspondances*. Nous écrivons la règle dans le fichier `regle.loxr`. Notre corpus de travail est `corpus.cnr`, le fichier propriété correspondant est `propriete.txt`. Le résultat du traitement se trouvera dans le fichier `res.loxf` et les informations relatives à ce traitement dans le fichier `inf.loxf`. La figure A.10 présente le fichier `arguments.lox` contenant les arguments correspondant à ce traitement. Ces arguments peuvent être spécifiés par l'intermédiaire de l'interface graphique de WinLoX. Pour réaliser le traitement avec DOSLoX, il suffit de taper la commande

```
doslox arguments.lox
```

dans l'invite de commande, mais avant cela, il faut écrire la règle correspondant à ce traitement.

Cette règle est représentée figure A.11. La MERA

```
[ems~"^V" & ems!~"^VINF" & ems!~"^VPAR"]
```

décrit un verbe conjugué (item 1). La MERS

```
[(ems~"^V" & ems!~"^VINF" & ems!~"^VPAR")
 & (lemme="être" | lemme="avoir")]
[ems!~"^(V|PCTFORTE)"]{0,4}
[ems~"^VPARP[^RES]"]
```

```
Requête :
(
  (
    [ems~"^V" & ems!~"^VINF" & ems!~"^VPAR"]
  )|(
    [(ems~"^V" & ems!~"^VINF" & ems!~"^VPAR")
     & (lemme="être" | lemme="avoir")]
    [ems!~"^(V|PCTFORTE)"]{0,4}
    [ems~"^VPARP[^RES]"]
  )
)
( [ems!~"^(V|PCTFORTE)"]{0,4} [ems~"^VINF"] ){1000,1}

Apparence : [B:[P:jeton][C:32];begin.index;end.index][C:10]
```

Figure A.11 – Règle `regle.loxr` correspondant au traitement de l'exemple.

décrit un auxiliaire conjugué (item 2) suivi de zéro à quatre mots ne contenant ni verbe ni ponctuation forte (item 3) suivi d'un verbe au participe passé (item 4). L'opérateur de disjonction `|` entre deux MER permet de faire la réunion entre les correspondances générées par la MERA et celles générées par la MERS. La MERS

```
[ems!~"^(V|PCTFORTE)"]{0,4}[ems~"^VINF"]
```

décrit des séquences de zéro à quatre mots ne contenant ni verbe ni ponctuation forte (item 5) suivis d'un verbe à l'infinitif (item 6). Le répétiteur préférentiel `{1000,1}` permet de répéter la séquence précédente un maximum de fois et au minimum une fois.

Le masque

```
[B:[P:jeton][C:32];begin.index;end.index]
```

est une boucle sur tous les mots de la correspondance (du premier, `begin.index` au dernier `end.index`) qui génère pour chacun des mots sa forme brute dans le corpus (`[P:jeton]`) suivis d'un espace (`[C:32]`). En fin de séquence, un retour chariot `[C:10]` est généré de manière à n'avoir qu'une séquence par ligne.

Pour donner une idée des temps de traitement : sur notre corpus de 6 468 522 mots occupant sur le disque une taille de 270 méga-octets, la durée de ce traitement sur un ordinateur équipé d'un processeur *AMD Athlon XP 1800+* est de trois minutes. Voici un petit extrait du fichier résultat :

- semblaient se concerter , se comprendre , échanger
- aurait voulu tout de suite se dévouer pour elle , la défendre , montrer
- pouvais point rester deux jours sans le faire venir
- dussent soulever et faire sauter le toit pour se répandre
- regarderaient faire le beurre , battre le grain , tondre les moutons , soigner
- permît de faire venir un peintre pour le dessiner
- pourrait prendre un sujet , épuiser les sources , en faire
- faut disloquer la phrase , souligner les mots , peser
- retenait pour compter les bouteilles , choisir des lattes , ou voir
- passait à écrire des lettres , à visiter les pauvres , à dissoudre des concubinages , à répandre
- avait des sourires pour saluer , pour répondre , pour approuver , pour remercier , pour prendre

- voudrais habiter un bel appartement , avoir des domestiques , bien manger , dormir
- semblait tout arranger , apaiser le père , faire rentrer
- goûtait une secrète joie à faire tomber les autres et à tirer
- est venu me voir pour me supplier de vous présenter
- devait finir par plaire à Maxime et lui paraître
- fit jurer de venir les rejoindre
- excuserai de courir les gueuses et de laisser se morfondre
- crut devoir aller siffler
- pourra non seulement s ' amuser , mais se loger , camper , faire ses courses , pratiquer un sport , se restaurer , participer à un congrès et travailler

Annexe B

Manuel d'utilisation du concordancier CooLoX

B.1 Présentation

CooLoX [1] est un puissant concordancier permettant de tirer parti de tout le pouvoir expressif des méta-expressions régulières pour définir la cible ou filtrer les contextes droit et gauche. Étant amené à interagir fortement avec l'utilisateur, CooLoX possède une interface graphique. Actuellement, CooLoX ne fonctionne que sur *Microsoft Windows*.

La cible des concordances étant définie par une méta-expression régulière, elle n'est absolument pas limitée à un seul mot (contrairement à d'autres concordanciers). Deux filtres permettent de préciser comment le contexte gauche doit finir ou comment le contexte droit doit commencer. CooLoX permet de sauver les concordances dans un fichier texte. CooLoX possède un module d'étiquetage permettant de réaliser de l'étiquetage vertical dans de bonnes conditions. Ce module peut très bien être utilisé pour effectuer des corrections ou des modifications dans le corpus.

B.2 Description de l'interface

B.2.1 Menu *Fichier*

Tous les paramètres du traitement en cours (chemin et paramétrage du corpus, méta-expressions régulières de la cible et des filtres sur les contextes, etc.) peuvent être enregistrés dans un fichier paramètre. Le menu *Fichier*, représenté figure B.1, permet essentiellement d'ouvrir (sous-menu *Ouvrir...*) et d'enregistrer (sous-menu *Enregistrer* et *Enregistrer sous...*) un fichier paramètre. Le sous-menu *Nouveau* permet de remettre tous les paramètres à leur valeur par défaut. Le sous-menu *Quitter* permet de quitter l'application.

1. Site Internet : `http://laurent.audibert.free.fr/lox.htm` . Référencement : `http://www.biomath.jussieu.fr/ATALA/outil/coolox_audibert_laurent.html` .

Figure B.1 – Menu *Fichier* de l'application CooLoX.

B.2.2 Menu *Corpus*

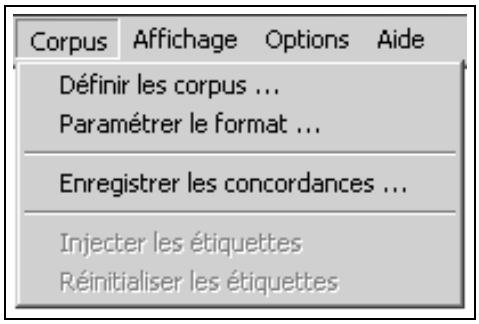


Figure B.2 – Menu *Corpus* de l'application CooLoX.

Le menu *Corpus* est représenté figure B.2.

Définir les corpus ... permet d'ouvrir une boîte de dialogue qui permet de :

- définir le chemin du ou des corpus de travail ;
- préciser le format du corpus (texte brut ou tabulaire) ;
- d'appeler la boîte de dialogue qui permet de préciser les paramètres des corpus tabulaires.

Cette boîte de dialogue est semblable au groupe *CORPUS TABULAIRE* de l'application WinLoX (cf. figure A.5).

Paramétrer le format ... permet d'appeler la boîte de dialogue qui permet de préciser les paramètres des corpus tabulaires. Cette boîte de dialogue reprend les fonctionnalités de la boîte de dialogue *Paramétrage du format des corpus tabulaires* de l'application WinLoX (cf. figure A.8).

Enregistrer les concordances ... permet d'enregistrer au format texte les concordances affichées.

Injecter les étiquettes permet d'enregistrer les étiquettes éditées à la main dans le corpus. Cet item n'est actif que quand CooLoX est en mode étiquetage (voir la section *Affichage* qui suit).

Réinitialiser les étiquettes permet d'ignorer toutes les étiquettes saisies.

B.2.3 Menu *Affichage* et sous-menu *Mode*

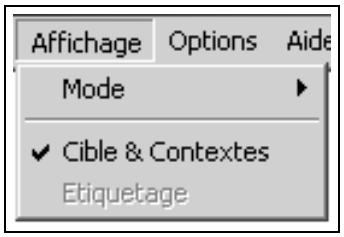


Figure B.3 – Menu *Affichage* de l'application CooLoX.

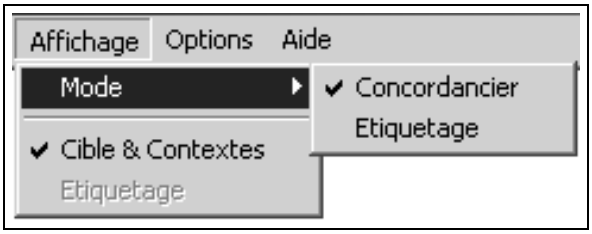


Figure B.4 – Menu *Mode* de l'application CooLoX.

Le menu *Affichage* est représenté figure B.3. Le sous-menu *Mode* de ce menu est représenté figure B.4.

L'item *Mode* permet de préciser si l'utilisateur désire travailler en mode *Concordancier* (item du sous-menu *Mode*) ou en mode *Étiquetage* (item du sous-menu *Mode*). Le mode *Étiquetage* permet de réaliser de l'étiquetage vertical, des corrections ou des modifications dans le corpus. À cet effet, ce mode ajoute une colonne de saisie dans le concordancier.

Lorsque *Cibles & Contextes* est coché, la boîte de dialogue *Cibles & Contextes* est affichée (cf. section B.2.7), sinon elle est masquée.

Lorsque l'item *Étiquetage* du menu principal est coché, la boîte de dialogue *Étiquetage* est affichée (cf. section B.2.8), autrement elle est masquée.

B.2.4 Menu *Options*

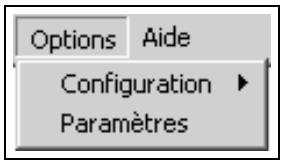


Figure B.5 – Menu *Options* de l'application CooLoX.

Ce menu, représenté figure B.5, permet d'enregistrer l'application dans la base de registre. L'intérêt de cette opération est l'association de l'extension du fichier paramètre avec l'application CooLoX ainsi que la prise en compte des icônes. Cette opération n'est utile que lors de la première utilisation de CooLoX sur une machine donnée.

B.2.5 Menu *Aide*

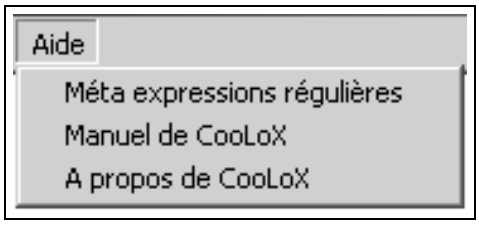


Figure B.6 – Menu *Aide* de l'application CooLoX.

Ce menu, représenté figure B.6, permet de visualiser :

- le fichier d'aide sur le formalisme des méta-expressions régulières et des masques, dont le contenu est proche de celui de la section 3.4 ;
- le fichier d'aide de CooLoX, dont le contenu est proche de celui de cette annexe ;
- les informations du type version de CooLoX, auteurs, contacts, etc.

B.2.6 Fenêtre principale

La figure B.7 est une copie d'écran de l'application CooLoX en mode *Concordancier*. Sur cette figure, les différentes zones importantes sont identifiées par des repères. La figure B.8 est une copie d'écran de l'application CooLoX en mode *Étiquetage*.

Sur la figure B.7 les repères 1, 2 et 3 représentent respectivement le contexte gauche, la cible et le contexte droit. Les entêtes de colonne 4, 5 et 6 permettent de trier dans l'ordre croissant ou décroissant les colonnes qu'ils identifient. Le tri sur la cible et le contexte gauche est un simple tri dans l'ordre lexicographique. Le tri sur le contexte gauche est également un tri dans l'ordre lexicographique, mais en considérant les mots dans le sens inverse de la lecture.

Le bouton 7 permet de réactualiser l'affichage. Lorsque l'affichage a besoin d'être réactualisé, celui-ci est grisé. C'est, par exemple, le cas lorsque l'un des paramètres de la boîte de dialogue *Cibles & Contextes* vient d'être modifié.

La fenêtre 11 affiche la cible de la ligne sélectionnée dans le concordancier. Cette cible est affichée en surbrillance accompagnée de son contexte. Le nombre de mots des contextes droit et gauche à afficher est précisé par la zone de saisie 9. Le bouton 10 permet d'effectuer un *copier* de la zone 11. La liste déroulante du repère 8 permet de préciser la forme des mots affichés dans la zone 11. Suivant les informations contenues dans le corpus, l'utilisateur peut observer les lemmes, les étiquettes morphosyntaxiques, etc.

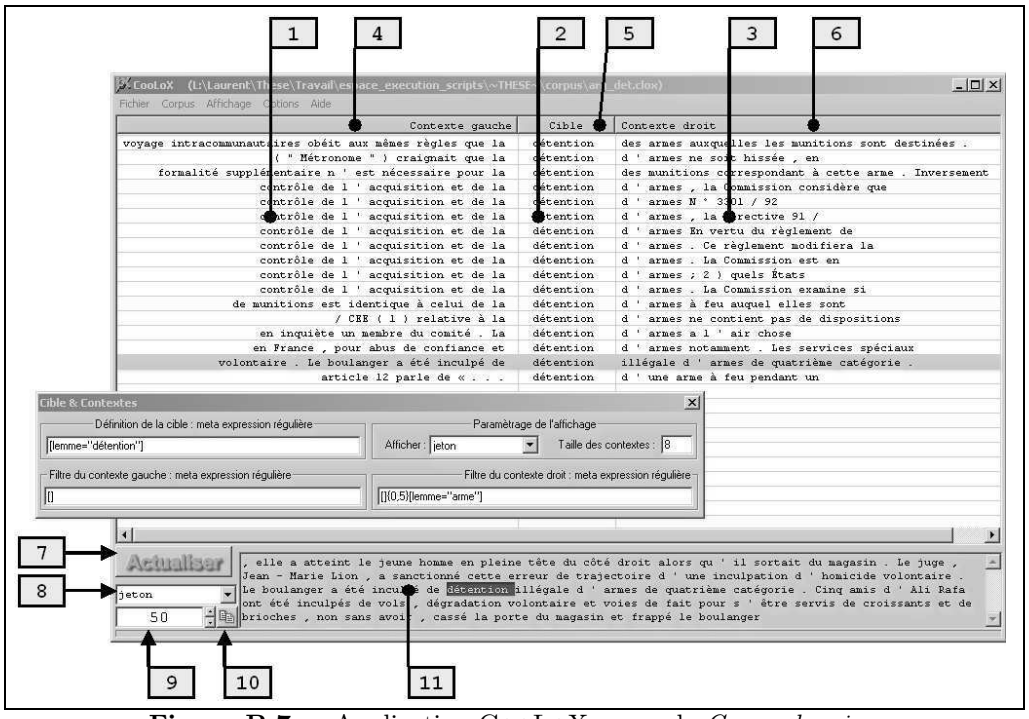


Figure B.7 – Application CooLoX en mode *Concordancier*.

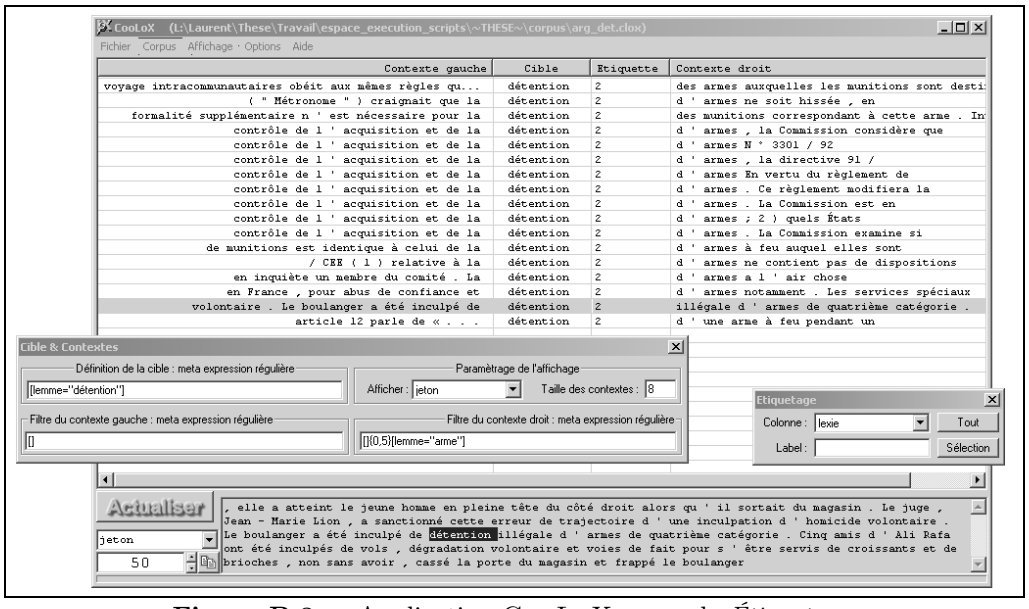


Figure B.8 – Application CooLoX en mode *Étiquetage*.

B.2.7 Boîte de spécification de la cible et des contextes

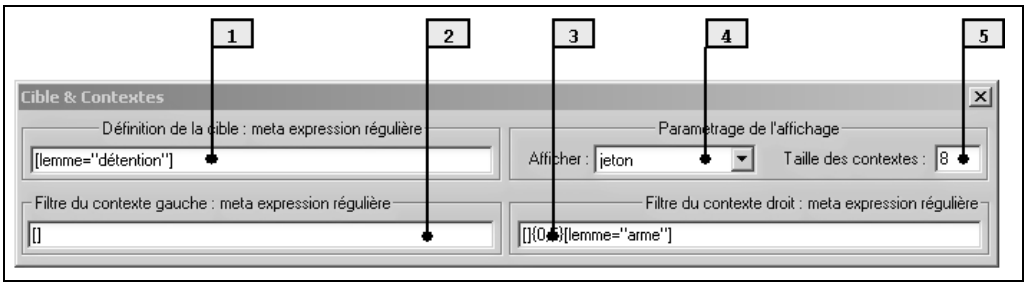


Figure B.9 – Boîte de spécification de la cible et des contextes de l'application CooLoX.

La figure B.9 est la boîte de dialogue de spécification de la cible et des contextes. Sur cette figure, les différentes zones de paramétrage sont identifiées par des repères. Cette boîte de dialogue est flottante.

La zone de saisie 1 permet de définir la cible. Cette définition se fait en écrivant une MER (cf. section 3.4.10). Les zones de saisie 2 et 3 permettent d'écrire des MER pour filtrer suivant le contexte droit et gauche. Les lignes qui sont affichées dans le concordancier respectent les trois conditions suivantes :

1. la portion de texte centrale (dans la colonne *Cible*) est décrite par la MER spécifiée dans la zone de saisie 1 ;
2. le début du contexte droit de la portion centrale est décrit par la MER spécifiée dans la zone de saisie 3 ;
3. la fin du contexte gauche de la portion centrale est décrite par la MER spécifiée dans la zone de saisie 2.

La recherche des portions de texte décrites par la MER spécifiée dans la zone de saisie 1 se fait en avançant dans le corpus mot par mot. Lorsque sur un mot donné, une ou plusieurs correspondances (commençant forcément par le mot courant) sont trouvées, seul la correspondance la plus grande est retenue. Nous procédons alors à une vérification des contextes droit et gauche pour savoir si la portion de texte doit être affichée ou pas dans le concordancier. La recherche recommence ensuite en se plaçant sur le mot qui suit le dernier mot décrit par la correspondance centrale.

La liste déroulante du repère 4 permet de préciser la forme que prennent les mots dans le concordancier. Suivant les informations contenues dans le corpus, l'utilisateur peut observer les lemmes, les étiquettes morphosyntaxiques, etc. Le nombre de mots des contextes droit et gauche à afficher dans le concordancier est précisé par la zone de saisie 5.

B.2.8 Boîte de saisie des étiquettes

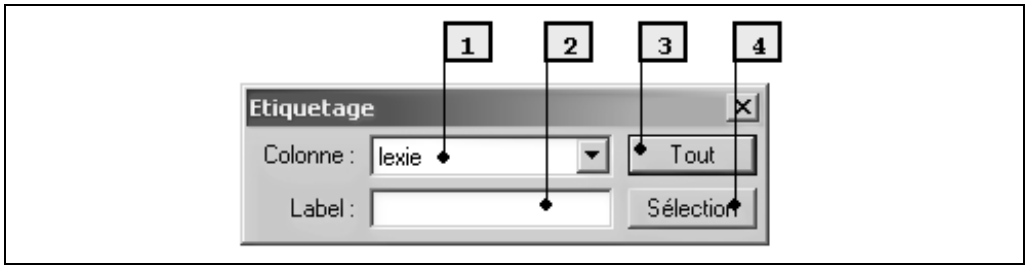


Figure B.10 – Boîte de saisie des étiquettes de l'application CooLoX.

La figure B.8 est une copie d'écran de l'application CooLoX en mode *Étiquetage*. La figure B.10 est la boîte de dialogue de saisie des étiquettes. Sur cette figure, les différentes zones d'interaction avec l'utilisateur sont identifiées par des repères. Cette boîte de dialogue est flottante.

La liste déroulante du repère 1 permet de préciser la propriété des mots sur laquelle l'utilisateur désire travailler. Dans le concordancier, la colonne *Etiquette* affiche cette propriété pour chaque cible.

En mode concordancier, il est préférable que la colonne *Cible* ne contienne qu'un mot par ligne. En effet, c'est une propriété de ce mot qui est éditée dans la colonne *Etiquette*. Si la colonne cible contient plusieurs mots, les modifications sont apportées au premier mot seulement. Les étiquettes peuvent être éditées individuellement dans la colonne *Etiquette*.

Si l'utilisateur désire affecter une même valeur à tous les mots ciblés par le concordancier, il peut saisir cette valeur dans la zone d'édition 2 et cliquer sur le bouton *Tout* (bouton 3). Une autre méthode de saisie par lots consiste à réaliser une sélection multiple de lignes dans le concordancier, puis à saisir la valeur commune dans la zone d'édition 2 et enfin à cliquer sur le bouton *Sélection* (bouton 4).

Annexe C

Exemples d'entrées du dictionnaire distributionnel

C.1 Avertissements et légende des entrées

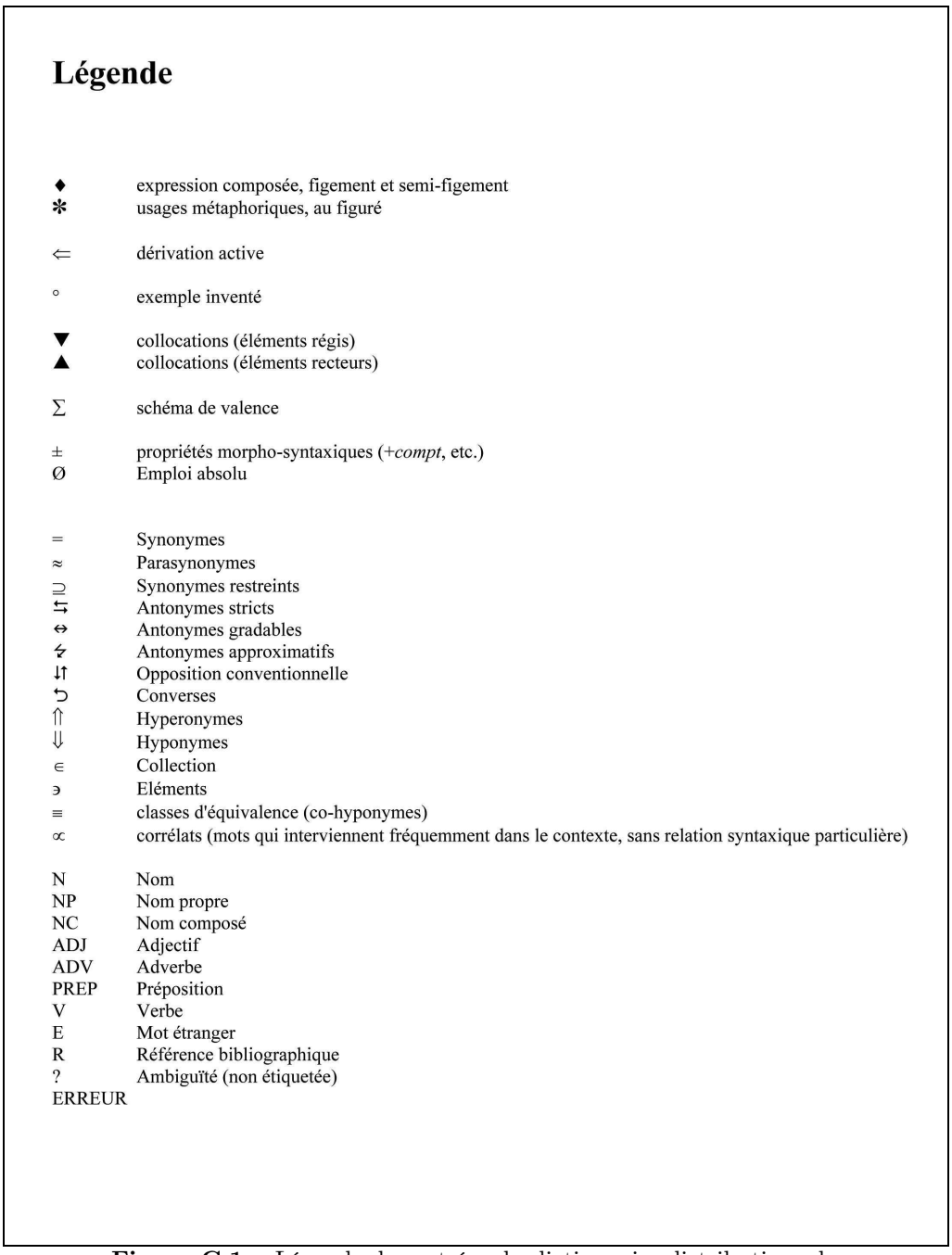
Nous faisons figurer ici quelques exemples d'entrées du dictionnaire distributionnel utilisé pour notre étude. Ces exemples ne sont donnés qu'à titre indicatif étant donné que ce dictionnaire fait l'objet d'un travail de thèse actuellement en cours.

En attendant d'être finalisée et portée vers un format plus malléable, la version préliminaire de ce dictionnaire a été réalisée sous *Microsoft Word*. Aussi, pour pouvoir faire figurer, sans les altérer, quelques exemples d'entrée dans ce mémoire au format LaTeX, les documents *Microsoft Word* ont été convertis en images avant d'être inclus.

Les entrées figurant dans les sections qui suivent sont :

- *barrage*, section C.2 page 267 ;
- *constitution*, section C.3 page 270 ;
- *détention*, section C.4 page 272 ;
- *pied*, section C.5 page 273 ;
- *clair*, section C.6 page 277 ;
- *frais*, section C.7 page 280 ;
- *utile*, section C.8 page 283 ;
- *arrêter*, section C.9 page 285 ;
- *comprendre*, section C.10 page 288 ;
- *importer*, section C.11 page 291 ;

La figure C.1 présente la légende de la signalétique utilisées pour les entrées.



Légende

♦	expression composée, figement et semi-figement
✻	usages métaphoriques, au figuré
⇐	dérivation active
°	exemple inventé
▼	collocations (éléments régis)
▲	collocations (éléments recteurs)
∑	schéma de valence
±	propriétés morpho-syntaxiques (+*compt*, etc.)
Ø	Emploi absolu
=	Synonymes
≈	Parasynonymes
⊇	Synonymes restreints
⇆	Antonymes stricts
⇔	Antonymes gradables
↯	Antonymes approximatifs
⇵	Opposition conventionnelle
⮌	Converses
⇑	Hyperonymes
⇓	Hyponymes
∈	Collection
∍	Eléments
≡	classes d'équivalence (co-hyponymes)
∝	corrélats (mots qui interviennent fréquemment dans le contexte, sans relation syntaxique particulière)
N	Nom
NP	Nom propre
NC	Nom composé
ADJ	Adjectif
ADV	Adverbe
PREP	Préposition
V	Verbe
E	Mot étranger
R	Référence bibliographique
?	Ambiguïté (non étiquetée)
ERREUR	

Figure C.1 – Légende des entrées du dictionnaire distributionnel.

C.2 Barrage

BARRAGE, *n. m.*

1. ⇐ BARRER

Def Action de barrer (gêner ou empêcher) physiquement le passage de quelqu'un, de quelque chose.

Ex °*le **barrage** de la rivière par les castors*
[Pas d'exemple dans le corpus]

✻ Empêcher ou gêner quelque chose. Barrage de quelque chose par quelque chose.
(J) *la représentation des minorités politiques de l'opposition, librement élues, avec toutes les formes démocratiques de **barrage** du système proportionnel, est indispensable pour garantir les droits fondamentaux du pluralisme.*

⇐ BARRER (Y BARRE X)

Σ BARRAGE DE X_{qqc} PAR Y_{qqx}

± *-compt, +sing*

2. ⇐ Ø

2.1. Barrage sur ...

Σ BARRAGE SUR qqc

± *+compt*

2.1.1. Barrage sur un fleuve, une rivière

Def Ouvrage qui interrompt le lit d'un cours d'eau, à des fins d'irrigation, de production d'énergie, de contrôle des eaux.

Σ BARRAGE SUR <*fleuve, rivière, ...*>

Ex *En septembre 1993, l'armée tuait quatre paysans de Madura qui manifestaient contre la construction d'un **barrage** sur leurs terres.*

*Le poisson réapparaît en nombre depuis trois ans, avec les "lâchages" du **barrage** d'Assouan.*

*Dans le cadre de cette renaissance du sud-est anatolien ... 22 **barrages** et 17 centrales hydroélectriques seront construits sur l'Euphrate et le Tigre.*

≈ DIGUE

⇑ OUVRAGE (HYDRAULIQUE)

⇓ BARRAGE-RÉSERVOIR, BARRAGE-DIGUE

≡ 1. PONT, DIGUE, DIGUETTE, CANAL, RETENUE

2. ROUTE, AUTOROUTE, CENTRALE, PONT, TUNNEL, LYCÉE, HÔPITAL, PORT, AÉROPORT

∝ AFFLUENT, AGRICOLE, AGRICULTURE, CULTURES, DÉFORESTATION, EAU, ÉCOLOGIE, ÉLECTRICITÉ, FLEUVE, FORÊT, INONDATION, IRRIGATION, IRRIGUER, MÈTRES CUBES, RETENUE, RIVIÈRE

▼
Adj grand, futur, gigantesque, coûteux barrage
barrage gigantesque, monumental, ultramoderne
haut barrage: *Le haut barrage d'Assouan*
barrage hydraulique, hydroélectrique
NP le barrage X : *le barrage Atatürk*
SPrep le barrage de X : *le barrage d'Assouan*
barrage sur un fleuve, une rivière : *le barrage sur le Danube*

▲
N le lac, les eaux, les turbines, les lâchages du barrage
le chantier, la construction, l'achèvement, le financement, l'inauguration, la rupture du barrage
un projet, un programme de barrage(s)
un système, une série de barrages
V(S) le barrage engloutit (des forêts, des bâtiments...), contrôle (le débit, les inondations), alimente (qqc en énergie, en électricité), fournit (de l'énergie, de l'électricité à qqc), se rompt, se brise
V(O) construire, édifier, financer, détruire un barrage
Prep en amont, en aval, en contrebas du barrage

✻ Usage métaphorique reprenant l'image d'un barrage qui se rompt (événement soudain et violent) ou qui déborde.

Ex *En se brisant soudainement, le **barrage** de la censure a libéré des cataractes, des avalanches d'interdits.*

(P) *le message de l'événement risque de déborder les **barrages** de commentaires visant à en atténuer la portée*

2.1.1.1. ♦ **BARRAGE-DIGUE, BARRAGE-RÉSERVOIR.**

2.1.2. Barrage sur une route

Def Blocage ou contrôle du passage sur une voie de communication, par quelqu'un ou un groupe de gens (forces armées, police, manifestants, etc.) à des fins de protestation, de contrôle, de coercition ou d'exactions.

Ex *Ces balles proviennent presque toujours de militaires énervés qui tirent pour un oui, pour un non, en particulier aux multiples **barrages** qu'ils ont établis sur toutes les routes encore praticables, pour racketter les gens.*

*La route du sud est hérissée de **barrages** policiers.*

*Dès les premières minutes qui suivirent le crash de l'avion, les équipes de tueurs dressèrent les **barrages** dans Kigali.*

Σ BARRAGE DE qqn SUR *<route, chemin...>*

≈ (POINT DE) CONTRÔLE

≡ 1. MANIFESTATION, OCCUPATION

2. CONTROLE, PATROUILLE

3. CONTROLES D'IDENTITE, ARRESTATION, FOUILLE, RAFLE, SAISIE, RACKET

▼	**Adj**	faux barrage barrages routiers, policiers, militaires barrage filtrant, volant
	SPrep	barrage de police, de l'armée, des douanes, de miliciens, de soldats, d'hommes en armes barrage sur une route, un chemin, une voie ferrée barrage de pierres

▲	**V(S)**	une route hérissée de barrages des barrages se dressent, foisonnent sur la route
	V(O)	dresser, établir, démanteler un barrage rencontrer, éviter, franchir un barrage

2.2. Barrage à, contre ...

2.2.1. Concret

Def Obstacle qui barre (gêne ou empêche) physiquement le passage de quelqu'un, de quelque chose.

Ex *La prudence conseille alors à toute puissance devant faire face à un éventuel **barrage** spatial antimissiles d'avoir plus, et non pas moins, de missiles nucléaires stratégiques*

*...le rôle de **barrage** humain assigné aux troupes d'Ankara, chargées de contenir une attaque en attendant que les armées modernes puissent monter au front.*

Σ BARRAGE À, CONTRE, ANTI- qqc

2.2.1.1. ♦ **BARRAGE D'ARTILLERIE**

Def Tir nourri contre une position ennemie, visant à empêcher sa progression, son passage.

Ex *...les Israéliens tirent argument de l'attentat contre leur ambassadeur à Londres pour lancer un **barrage d'artillerie** sur le Liban sud*

▲	**V(0)**	lancer un barrage d'artillerie sur qqx.

2.2.2. Abstrait

Def Obstacle ou difficultés créés par quelqu'un pour gêner, empêcher, faire échouer quelque chose.

Ex *Chacun souhaite, sans toujours y croire, que M. Kim Jong-il impose son pouvoir (évolutif), seul **barrage** à un chaos généralisé.*

*L'habitude de consommer une bière le plus souvent produite localement, avec sa saveur spécifique, constituera un **barrage** contre les breuvages au goût standardisé*

*C'est un choix définitif qui ancre [la Turquie] au continent, sans retour possible, un **barrage** face aux forces antieuropéennes à l'intérieur.*

Σ BARRAGE DE qqn À/CONTRE/FACE À qqc

= OBSTACLE

≈ DIFFICULTÉS

2.2.2.1. ♦ **FAIRE BARRAGE**

Def Créer un obstacle ou des difficultés de façon à gêner, empêcher, faire échouer quelque chose.

Ex *L'Irak **ferait barrage** aux éventuelles entreprises perturbatrices de l'Iran*

Σ FAIRE BARRAGE A qqx

= FAIRE OBSTACLE

≈ FAIRE ÉCHOUER, EMPÊCHER, GÊNER, BARRER LA ROUTE, CRÉER DES DIFFICULTÉS

↯ SOUTENIR, DONNER UNE CHANCE

3. ♦ **TIR DE BARRAGE**

1. Propre

Def Tir nourri contre une position ennemie, visant à empêcher sa progression, son passage.

Ex *Aussitôt, le **tir de barrage** allemand redoubla. Les mitrailleuses crépitaient. (web)*
[Pas d'exemple dans le corpus]

Σ TIR DE BARRAGE DE qqx SUR qqx

≈ SALVE, RAFALE

2. Figuré

Def Succession d'attaques ou de critiques contre quelqu'un, une idée, un projet, etc.

Ex *C'est donc, à quelques exceptions près, à un véritable **tir de barrage** que s'est heurté le livre de Günter Grass.*

*En dirigeant leur **tir de barrage** contre la convention sur la biodiversité, les États-Unis défendent leurs intérêts économiques...*

*Malgré les **tirs de barrage** et les provocations de la fraction la plus réactionnaire ... la tendance "modernisatrice" toujours dominante...*

Σ TIR DE BARRAGE DE qqn CONTRE qqx

▼	**Adj**	un véritable tir de barrage
▲	**V(O)**	diriger un tir de barrage sur/contre qqx se heurter, être soumis à un tir de barrage

C.3 Constitution

CONSTITUTION, *n. f.*

1. La Constitution

Def Ensemble de lois qui établissent la forme d'un gouvernement

Ex *Article 5 de la* ***Constitution***

▼	**Adj**	*Nouvelle, ancienne, actuelle Constitution*

▲	**N**	*Amendement, révision, modification de la Constitution* *Article x de la Constitution* *Projet de Constitution*
	V(O)	*Amender, réviser, modifier, adopter, rédiger, respecter la Constitution* *Etre contraire à la Constitution*
	Sprep	*Inscrire dans la Constitution*
	Ppassé	*Reconnu, stipulé, affirmé, prévu, garanti par la Constitution*

⇐ Ø

Σ N DE CONSTITUTION

± *+ majuscule*

± *- animé*

1.1. ♦ CONSTITUTION CIVILE DU CLERGÉ

Ex (A) *le janséniste Camus , qui rédigeait la* ***constitution*** *civile du clergé, croyait aux miracles du diacre Pâris*

2. ⇐ Etre constitué

Def Regrouper des éléments pour former un tout

Ex *la* ***constitution*** *de réserves de bois*

(M) *la* ***constitution*** *des listes et des alliances politiques*

▲	**Sprep**	*En voie, en cours de constitution*

⇐ X EST CONSTITUÉ (PAR Yqqn)

Σ LA CONSTITUTION DE Xqqc (PAR Yqqn)

± *+sing*

≈ FORMATION, CRÉATION, FONDATION

3. = composition

Ex (O) *c ' est bien un engin nouveau , mais plus par son moteur que par sa* ***constitution*** *générale*

(A) *la* ***constitution*** *moléculaires de solides*

(A) *la* ***constitution*** *de la matière*

⇐ Ø

Σ CONSTITUTION DE Xqqx <*substance, matière, objet…*>

± *- animé, + sing*

= COMPOSITION

4. + animés humains

Def Complexion du corps humain

⇐ Ø

Σ CONSTITUTION DE Xqqn

± *+ animé humain, + sing*

Ex (O) *« Si donc en procédant à l ' examen médical [. . .] , le médecin - juge estime que le postulant ou la postulante est condamné , de par sa **constitution** physique, à ne mettre au monde que des enfants dégénérés , il pourra [. . .] prononcer une sentence de stérilisation*

▼	**Adj**	*Une constitution robuste*

5. ♦CONSTITUTION DE PARTIE CIVILE

Def Demande de réparation formée devant un tribunal pénal par une personne qui se prétend victime d'une infraction

Ex (O) *la victime pourra désormais déclencher les poursuites pénales non seulement par la voie de la citation directe, mais également par le biais d'une plainte assortie d'une **constitution de partie civile** déposée entre les mains du juge d'instruction .*

⇐ X_{QQN} SE CONSTITUE PARTIE CIVILE

Σ CONSTITUTION DE PARTIE CIVILE

∈ JUSTICE

∝ JUGE, INSTRUCTION, TRIBUNAL, POURSUITES

▲	**N**	*Plainte avec constitution de partie civile*

E.

Ex (A) *Molecular **constitution** of Matter , § 29 44 (Proceedings of the Royal Society of Edimburgh , 1er et 15 juillet 1889 ; Scientific Papers , VOL III , p . 404)*

C.4 Détention

DÉTENTION, *n. f.*

1. Détention de qqn

Def Fait d'être incarcéré ou enfermé

⇐ X EST DÉTENU QUELQUE PART

Σ DÉTENTION DE X_{QQN} QUELQUE PART

Ex (M) *Gracié après vingt - quatre ans de* ***détention****, Seznec reviendra du bagne en juillet 1947 , brisé , semblable à une ombre .*

± *- compt, + animé*

▼	**Adj**	*Détention provisoire, préventive*
	Sprep	*Détention en < pays>, à <lieu>*

≈ EMPRISONNEMENT, INCARCÉRATION

∝ PRISONNIER, ARRESTATION

▲	**N**	*Condition, lieu, centre de détention* *Durée, x< jour, mois, année> de détention*
	V(O)	*Etre, placer, maintenir, mettre en détention*

2 Détention de qqch

Def Avoir en sa possession

⇐ Y DÉTIENT QQCH

Σ DÉTENTION DE X_{QQC} PAR Y_{QQN}

DETENTION DE *<objet, animal…>*

± *+compt, - animé*

Ex (J) *les conditions régissant la* ***détention*** *des lévriers et leur participation à des courses relèvent des autorités locales , et non de la Commission*

(M) *En 1983, la* ***détention*** *desdits stupéfiants pour usage personnel a été légalisée*

▼	**Adj**	*Détention illégale*
	Sprep	*Détention d'armes (à feu)*

= POSSESSION

∝ ACQUISITION

C.5 Pied

PIED, *n. m.*

1. ⇐ Ø

⇐ Ø

± *+ compt, - animé*

1.1. Le pied de qqx

1.1.1. Le pied de < *animé*>

Σ PIED DE <*humain, animal*>

Def Partie de l'extrémité de la jambe qui sert à l'homme à se soutenir et à marcher.

Ex (A) *Il sentit un **pied**, un petit **pied**, qui rôdait sous la table.*

⇑ PARTIE DU CORPS

⇓ MAIN, POING, TÊTE

∝ CHAUSSURE

▼	**Adj**	*Nu-pieds, pieds nus, plats*
	Sprep	*Mettre pied à terre*

▲	**N**	*La plante, pointe, les ongles des pieds* *Coup de pied*
	V(O)	*Frapper des pieds* *Poser les pieds qqpart*
	Prep	*Sous les pieds*

✻ (M) *Mais, parallèlement, il a un **pied** dans l'Orchestre national de Radio - France, où il est chargé des " événements spéciaux ".*

✻ (A) *Las ! s'entre - disaient les bourgeois, voici que le duc d'Anjou a le **pied** en nos pays.*

1.1.1.1. ♦ METTRE LES PIEDS QUELQUE PART

Def Y aller, y passer.

Ex (A) *Il avait déclaré d'abord qu'il n'irait point à la fête du Patron, et qu'il ne voulait plus **mettre les pieds** chez ce sale juif.*

1.1.1.2. figuré

Def Manière d'agir

Ex (A) *Il rentra chez lui d'un **pied** joyeux.*

▼	**Adj**	*De pied ferme*

1.1.2. ✻ Le pied de <*objet*>

Σ PIED DE <*meuble…*>

Def Partie d'un objet servant de support.

Ex (A) *La petite table a les quatre **pieds** coupés*

*(A) …et des mousses rongeaient le **pied** des colonnes de plâtre, tandis que des herbes folles avaient disjoint la chaux des frontons.*

1.1.3. Le pied de <*végétal*>

Σ PIED DE <*arbre…*>

Def Partie du tronc ou de la tige d'un végétal qui est le plus près du sol.

Ex (A) *Le **pied** des arbres s'amincit jusqu'à disparaître.*

1.1.3.1. Un pied, des pieds de <*végétal*> Σ UN PIED DE <*vigne…*>

Def Désigne l'unité

Ex (A) *Un pied de **Vanille**, dont les grosses gousses mûres exhalaient des senteurs pénétrantes, courait sur la rondeur d ' un portique garni de mousse.*

1.1.3.2. ♦ **SUR PIED**

Ex (A) *Plusieurs fois, dans mes rêves, j'ai inventé le dendromètre, appareil à mesurer les arbres **sur pied**.*

1.2. Au, du pied de qqx

⇐ Ø

± *- compt, - animé*

Def Partie inférieure d'une chose élevée.

1.2.1. Au pied de <*animé*> Σ AU PIED DE <*humain, animal…*>

Ex (A) *Le cri Lantenac ! éclata à son oreille, et à ses **pieds**, à travers les ronces et les branches, des faces violentes apparurent.*

(A) *Mais il s'était assis sur le coin du divan, presque aux **pieds** de la jeune femme.*

▲	**V**	*S'accroupir, s'asseoir, s'étendre, se jeter, s'agenouiller, tomber, se coucher au(x) pied(s) de qqn*

1.2.2. Au pied de quelque part Σ AU PIED DE <- *animé*>

Ex (A) *Le docteur parvint au pied de l'échelle.* ≈ EN BAS

1.2.3. Abstrait (1 occ.) Σ AU PIED DE <- *animé*>

Ex (M) *Les titres de huit ans de durée sont rémunérés au taux d'intérêt facial de 9, 25 %, ce qui, au **pied** des commissions (de 2 % pour les banques), correspond à un rendement de 9, 32 %.*

1.3. ♦ A pied

± *- compt, - animé, + sing*

1.3.1. Être à pied Σ X_{QQN} À PIED

Def Marcher. Ne pas être transporté par un véhicule ou une monture.

Ex (M) *Les deux garçons ont donc décidé de rentrer chez eux **à pied**, par la route.*

▲	**N**	*Marche à pied*
	V	*Aller à pied qqpart*

1.3.2. N à pied Σ X_{QQCH} À PIED

Def Indique la forme.

Ex (A) *Il avait un pantalon **à pied**, des pantoufles, un gilet qui semblait avoir été de satin blanc…*

° verre à pied

1.4. Figements et semi-figements

1.4.1. ♦ METTRE SUR **PIED** QQCH, LA MISE SUR **PIED** DE QQCH

1.4.2. ♦ METTRE À **PIED** QQN, LA MISE À **PIED** DE QQN

1.4.3. ♦ METTRE SUR LE (MÊME) **PIED**

1.4.4. ♦ (METTRE) SUR UN **PIED** D'ÉGALITÉ, ÊTRE SUR UN MÊME **PIED**

1.4.5. ♦ (PRENDRE) AU **PIED** DE LA LETTRE

1.4.6. ♦ À **PIED** D'ŒUVRE

1.4.7. ♦ ARME AU **PIED**

1.4.8. ♦ AU PETIT **PIED**

1.4.9. ♦ AU **PIED** !

1.4.10. ♦ AVOIR UN **PIED** DANS LA TOMBE

1.4.11. ♦ CASSER LES **PIEDS**

1.4.12. ♦ COUPER L'HERBE SOUS LE **PIED**

1.4.13. ♦ D'UN BON **PIED**

1.4.14. ♦ DE **PIED** EN CAP (=des pieds à la tête)

1.4.15. ♦ DES **PIEDS** À LA TÊTE, DE LA TÊTE AUX **PIEDS**

1.4.16. ♦ ÊTRE **PIEDS** ET POINGS LIÉS

1.4.17. ♦ ÊTRE SUR LE **PIED** DE GUERRE

1.4.18. ♦ FAIRE LE **PIED** DE GRUE

1.4.19. ♦ FOULER AU **PIED** QQCH

1.4.20. ♦ GARDER, AVOIR LES **PIEDS** SUR TERRE

1.4.21. ♦ LE BONHEUR À SES **PIEDS**

1.4.22. ♦ METTRE LE **PIED** À L'ÉTRIER

1.4.23. ♦ METTRE LES **PIEDS** DANS LE PLAT

1.4.24. ♦ NE PLUS SAVOIR SUR QUEL **PIED** DANSER

1.4.25. ♦ **PIED** À **PIED**

1.4.26. ♦ **PIED** DE NEZ

1.4.27. ♦ **PIED** DE PAGE

1.4.28. ♦ UN **PIED**-NOIR

1.4.29. ♦ PORTRAIT EN **PIED**, EN **PIED**

1.4.30. ♦ PRENDRE SON **PIED**, C'EST LE **PIED** !

1.4.31. ♦ REMPLACER AU **PIED** LEVÉ

1.4.32. ♦ RETOMBER SUR SES **PIEDS**

1.4.33. ♦ TROUVER CHAUSSURE À SON **PIED**

1.4.34. ♦ (UN GÉANT, COLOSSE) AU **PIED** D'ARGILE

1.4.35. ♦ VIVRE SUR UN GRAND **PIED**, SUR LE **PIED** DE

1.4.36. ♦ **PIED** AU PLANCHER

1.4.37. ♦ **PIED**-À-TERRE

1.4.38. ♦ **PIED**-TENDRE

1.4.39. ♦ (REMETTRE) SUR **PIED** QQN

1.4.40. ♦ VALET DE **PIED**

1.4.41. ♦ BON **PIED**, BON OEIL

1.4.42. ♦ (FAIRE QQCH) COMME UN, DES **PIED**(S)

1.4.43. ♦ MARCHER SUR LES **PIEDS** DE QQN

1.4.44. ♦ AVOIR LE **PIED** MARIN

1.4.45. ♦ LEVER LE **PIED** (ralentir)

1.4.46. ♦ AVOIR LE MONDE À SES **PIEDS**

1.4.47. ♦ (RE)PRENDRE **PIED**, DANS, SUR

1.4.48. ♦ TRAÎNER LES **PIEDS**

1.4.49. ♦ APPELS DU **PIED** (fig.)

1.4.50. ♦ (ÊTRE) AU **PIED** DU MUR

1.4.51. ♦ PERDRE **PIED**

1.4.52. ♦ LACHER **PIED**

1.4.53. ♦ (PLONGER) À **PIEDS** JOINTS DANS

1.4.54. ♦ (PRENDRE) À, LE, EN CONTRE-**PIED**

1.4.55. ♦ DE PLAIN-**PIED**

1.4.56. ♦ UN CROCHE-**PIED**

1.4.57. ♦ DES VA-NU-**PIEDS**

1.4.58. ♦ UN MARCHE-**PIEDS**

1.4.59. ♦ UN CHAUFFE-**PIEDS**

1.4.60. ♦ ÊTRE À TANT DE **PIED** SOUS TERRE

1.4.61. ♦ CHANGEMENT DE **PIED** (D'AVIS ?)

2. ⇑ Mesure

⇐ Ø

2.1. Distance

Σ $X_{NUMÉRAL}$ PIED

± *+ compt, + adj. numéral, - animé*

Def Ancienne mesure de longueur valant 33cm. ⇑ MESURE

Ex (A) *Dans cinq minutes, nous serons à cinquante **pieds** du sol.* ⇓ POUCE

≡ LIVRE

▼	**Sprep**	*X pied de long, de haut*

▲	**N**	*Longueur, largeur, diamètre de x pied*

2.1.1. ♦ **PIED CUBES**

2.2. Le pied de <*vers*> (1occ.)

Σ UN VERS DE $X_{NUMÉRAL}$ PIED

Def Groupe de syllabes constituant la mesure élémentaire du vers. ⇑ MESURE

Ex (A) *J'ai horreur de la rime, surtout en prose.*
La justice est gratuite. Heureusement, elle n'est pas obligatoire.
Je me sens « Jumeaux ».
*Un beau vers a douze **pieds**, et deux ailes.*

∝ PROSE, RIME

NP.

Ex (M) *Ou alors suivons tout de suite les conseils prodigués par le célèbre **Pied-Nickelé** Croquignol…*

(M) *Les **Casse-Pieds** ont longtemps joué dans le métro, une partie des musiciens sont allés rejoindre la Mano Negra.*

C.6 Clair

CLAIR, E, *adj.*

1. ↯sombre, obscur

1.1. ≈ lumineux, éclatant

Def Qui a de l'éclat, qui est lumineux

⇐ LA CLARTÉ DE QQCH

Σ N CLAIR, CLAIR N

Ex (A) *Le lendemain par un **clair** soleil, la procession sortit de l'église*

∝ OMBRE, LUMIÈRE, VISIBLE, CIEL

▲	N	*Clair soleil*

1.2. <*endroit*> clair

Def Recevant de la lumière

⇐ LA CLARTÉ DE <*endroit, lieu*>

Σ N CLAIR, CLAIR N

Ex (1occ.) (M) *Dans son vaste bureau **clair** de l'hôtel de Bourgogne*

1.3. ↯ foncé

Def De couleur non foncée

⇐ Ø

Σ N CLAIR, CLAIR N

Ex (A) *Il était poudré, ganté, brossé, boutonné ; son habit bleu **clair** ne faisait pas un pli.*

▲	N	*Yeux, regards clairs* *Gris, bleu clair*

1.4. Qui laisse passer la lumière

1.4.1. ≈ Transparent, pur

⇐ LA CLARTÉ DE <*eau*>

Σ N CLAIR, CLAIR N

Ex (M) *Malgré une légende qui court de vallon en vallon, par-dessus les torrents **clairs** du Caldas, et dans la vallée tropicale, oppressante et chaude comme l ' enfer du Magdalena.*

▲	N	*Eau claire*

∝ FRAIS

1.4.2. ≈ Dégagé, sans nuages

⇐ LA CLARTÉ DE <*ciel, jour*>

Σ N CLAIR, CLAIR N

Ex (A) *Par un jour de juin, **clair** et doux, fut dressé à Bruxelles, sur le marché devant la Maison de Ville, un échafaud couvert de drap noir et y attenant deux poteaux élevés, garnis de pointes de fer.*

▲	N	*Jour, nuit clair(e)* *Ciel clair* *Temps clairs*

∝ DOUX, FRAIS

1.5. ↯ dense, consistant

Def Peu dense, sans beaucoup de matière.

⇐ Ø

Σ N CLAIR, CLAIR N

Ex (2occ.) (A) *Du vin blanc ou du rouge, poulet aux champignons, radis blanc et beurre, puis des pigeons dans un jus si **clair** qu'ils ressemblent à des poules d'eau.*

1.6. <*partie du corps*>

Ex (A) *Le feu de la cuisine eût noirci son teint **clair** comme le jour.*

⇐ Ø
Σ N CLAIR, CLAIR N
∝ CORPS, YEUX

▲ **N** *Visage clair*

1.7. ≈ Net, distinct <*forme*>

Ex (1occ.) (A) *Elle était toute gonflée de volupté, et les lignes **claires** de ses épaules et de ses reins se détachaient avec des sécheresses félines sur la tache d'encre dont la fourrure noircissait le sable jaune de l'allée.*

⇐ Ø
Σ N CLAIR, CLAIR N
± *- ante*

2. La clarté de <*propos, situation…*>

2.1. ≈ Compréhensible, ↯ vague

Def Facilement intelligible.

⇐ LA CLARTÉ DE <*document, propos*>
Σ N CLAIR, CLAIR N
∝ SIMPLE, CORRECT, PRÉCIS

Ex (P) *On observe alors que le modèle syllogistique, si net, **clair** et indiscutable, est la forme la plus achevée de la rhétorique.*

▼ **Adv** *Très, tout à fait, aussi, assez clair*

▲ **N** *Propos, phrase, texte clair*

2.1.1. ♦ DIRE QQCH HAUT ET CLAIR

[non observé dans le corpus]

2.2. ≈ Manifeste, évident

Def Manifeste, évident, pour une pensée, un propos.

⇐ LA CLARTÉ DE <*pensée, propos*>
Σ N CLAIR, CLAIR N
∝ DISTINCT, FERME

Ex (P) *Le constat est **clair**. Malgré le spectacle, les émissions politiques ont dû décrocher du début de soirée.*

▼ **Adv** *Très, moins clair*

2.3. ♦ Il…clair que

Ex (O) *S'**il est clair que** « dans un jeu à somme nulle, j'ai intérêt à garder le secret sur la stratégie que j'adopte », on peut dire que « c'est exactement le contraire pour les jeux mixtes…*

⇐ Ø
Σ IL…CLAIR QUE

▼ **Adv** *Il est bien, donc, assez, très clair que*

▲ **V** *Être, paraître, apparaître, devenir clair que*

2.4. = intelligent

Def Se dit d'une personne intelligente.

⇐ Ø

Σ N CLAIR

Ex (1 occ) (A) *Jaurès dit […] De Shakespeare :*
*-- Il est plus latin, plus **clair**, qu'on ne croit. Hamlet n'est pas un homme profond, mais un pauvre jeune homme accablé par le poids de ce qu'il ne peut pas faire.*

± *- ante, + animé humain*

2.5. Être clair sur quelque chose

Def Ne rien cacher.

⇐ LA CLARTÉ DE X SUR Y

Σ ÊTRE CLAIR SUR X_{QQCH}

Ex (1 occ) (M) *Si j'ai parlé fort, c'est qu'on avait déclenché une offensive contre l'OM, alors que c'est le club le plus **clair** sur ses comptes.*

3. ≈ Net, cristallin <*son*>

Ex (M) *Il a le regard posé, une élocution calme et **claire**.*

⇐ LA CLARTÉ DE X

Σ <*son*> CLAIR

∝ RIRE, HAUTE, TON

▲ **N** *Voix claire*

4. Figements et semi-figements

4.1. ♦ DES COUPES **CLAIRES** DE QQN

4.2. ♦ C'EST **CLAIR**

4.3. ♦ ÊTRE ON NE PEUT PLUS **CLAIR**

4.4. ♦ (Y) VOIR **CLAIR** (ADV.)

4.5. ♦ EN **CLAIR** (SUBS)

4.6. ♦ LE PLUS **CLAIR** DE (LOC.)

N.

Ex (A) *il gelait terriblement par un **clair de lune** limpide.*

?.

Ex (A) *Lamme en fièvre était bien attaché sur son lit […] et, se croyant encore en cuisine, il disait :*
*-- Ce fourneau est ***clair** aujourd'hui. Tantôt il pleuvra des ortolans.*
(* vide ?)

Ex (M) *On peut y voir un effet du temps, un état du jazz, le retour à la forme **claire**, aux arrangements bien troussés, une autre histoire.*

C.7 Frais

FRAIS, FRAÎCHE, *adj.*

1. ≈ Froid

Def Qui est légèrement froid ou qui donne une sensation de froid modéré.

⇐ LA FRAÎCHEUR DE X

1.1. ≈ Glacial ↯doux, torride

Σ <*endroit, air*> FRAIS, FRAIS N

Ex (A) *A midi, par un clair soleil et un vent **frais**, les chariots s'en furent verdoyants et fleuris…*

(P) *il ressort que l'animal préfère les zones **fraîches** et ombragées…*

≈ FROID

∝ CLAIR, GLACE, TEMPÉRATURE

▲	**N**	*Air, vent frais*

1.1.1. ♦ FAIRE FRAIS (adv)

Ex (1 occ) (A) *Mes beaux pères, dit-il, il **fait frais**, je suis peu vêtu, vous l'êtes trop.*

≈ FRISQUET

↯ BON, CHAUD

1.2. ≈ glacé ↯ brûlant

Σ N FRAIS, FRAIS N

Ex (A) *Tu as bien parlé, dit Ulenspiegel, maintenant il faut bien boire. - - J'ai encore la langue **fraîche**, dit la fille.*

(A) *Elle ne sait pas qu'il y a là-haut des bains suaves, pleins de parfums, où roulent de grands morceaux de sucre candi blanc et **frais** comme glace.*

(M) *des serveurs en nœud papillon proposent des sandwiches libanais et des boissons **fraîches**….*

▲	**N**	*Eau, boisson fraîche*

1.2.1. ♦ BOIRE FRAIS (adv)

Ex (2 occ) (A) ***Boire frais** et guerroyer salé, c'était écrit, dit Ulenspiegel.*

2. ≈ Pur ↯pollué

Def Qui n'est pas pollué

⇐ Ø

Σ <*air*> FRAIS

Ex (A) *Il alla ouvrir sa fenêtre pour avaler une bonne tasse d'air **frais**….*

± *- ante*

(A) *Un parfum léger s'envolait du peignoir, le parfum **frais** de la toilette récente.*

▲	**N**	*Air frais*
	V	*Humer l'air frais*

3. ↯ Vieux

3.1. ≈ Neuf, récent ↯ ancien

⇐ Ø

Σ N FRAIS, FRAIS N

Ex (A) *ils ne prirent pas garde à de larges piétinements, à des traces **fraîches** qui marquaient çà et là le sol humide.*

3.1.1. ♦ FRAIS ÉMOULU

Ex (M) *un jeune capitaine français **frais émoulu** de Coëtquidan…*

3.1.2. ♦ DE FRAÎCHE DATE

3.1.3. FRAIS [ADV] (+ P.PASSÉ)

Ex (M) Louis Chavance adapte et dialogue avec Henri-Georges Clouzot, cinéaste tout **frais révélé**, son scénario…

3.1.4. ♦ ARGENT FRAIS

3.2. ↯ Fané, sec, flétri

⇐ Ø

Σ N FRAIS, FRAIS N

3.2.1. - animé

Ex (A) *Le salon, tapissé de papier ramagé, assez **frais**…*

3.2.2. *<fleurs, végétaux>*

Ex (A) *nous avons le temps de vous entendre, dit Joe en s'étalant voluptueusement sur l'herbe **fraîche***

(J) *Il est vrai que la floriculture communautaire, et spécialement le secteur des fleurs coupées **fraîches**, est soumise à une concurrence accrue…*

3.2.3. + animé humain

≈ JEUNE

↯ MÛR

Ex (A) *« C'est vraiment un joli parti pour vous, si jolie, si **fraîche**, et si intelligente. »*

(A) *Je n'ai plus rien de moi, de moi l'homme radieux, **frais** et fort que j'étais à trente ans.*

3.3. *<partie du corps, expressions>*

⇐ Ø

Σ N FRAIS, FRAIS N

Ex (A) *C'est ton sourire, Nele, **frais** comme le matin, doux comme le rayon.*

(A) *ma mignonne femme aux cheveux d'un brun doré, aux yeux bruns, aux joues **fraîches**…*

4. ≈ Pur

4.1. <*voix, son*> ≈ clair, net, aigu

⇐ Ø
Σ N FRAIS, FRAIS N
∝ CLAIR

Ex (A) *c'étaient comme des chants d'ange, toutes ces voix **fraîches** montant vers le ciel.*

(A) *Ces voix montaient claires et **fraîches** jusqu'au vieillard pensif.*

4.2. <*animé humain, sentiments*> ≈ innocent, ingénu

⇐ Ø
Σ N FRAIS, FRAIS N
∝ ENFANT

Ex (A) *il semble qu ' un parfum sorte de ces **fraîches** âmes.*

(A) *Lamme alors considérant la fillette, dit : - - Je te vois **fraîche**, embaumée, ton épaule sortant de ta robe comme une grande feuille de rose blanche.*

5. ≠en conserve, rassis, avarié

Def Nouvellement produit ou récolté. Qui n'est pas altéré.

⇐ Ø
Σ <*aliment, nourriture*> FRAIS
± *- ante*

Ex (A) *… tu nous approvisionneras de viande **fraîche**.*

(J) *…la vente directe soit à l ' état **frais**, soit après transformation, à des détaillants ou au consommateur.*

▲	N	*Viande fraîche* *Poisson, œuf, légume, produit frais*

∝ CONGÉLÉ, RÉFRIGÉRÉ

***** (A) *Chaque matin Renée prenait un bain de quelques minutes. Ce bain emplissait pour la journée le cabinet d ' une moiteur, d'une odeur de **chair fraîche** et mouillée.*

6. ≈ Reposé, en forme≠fatigué

Def Qui a conservé ou recouvré ses forces, sa vitalité.

⇐ Ø
Σ <*personne, animal*> FRAIS
± *- ante,+ animé*

Ex [pas d'exemple dans le corpus]

6.1. ♦ FRAIS ET DISPOS

Ex (A) *Tu restes donc ici, **frais**, **dispos**, alerte comme jeune aigle.*

N.

Ex (O) *…réduire les **frais** de 40%…*

Ex (A) *Le gilet de Philippe tient au **frais** la cruche d'eau.*

?.

Ex (A) *L'intérieur d'un cochon est **frais** comme le trousseau d'une mariée.*

C.8 Utile

UTILE, *adj.*

1. - Animé

Def Dont l'usage, la pratique, est ou peut-être avantageux pour quelqu'un ; qui satisfait un besoin, répond à une demande sociale.

⇐ L'UTILITÉ DE X_{QQCH}

Σ N UTILE, UTILE N

▲	**V**	*Demeurer, devenir, se révéler, considérer comme, s'avérer, juger, rendre utile*
	Adv	*plus, moins, bien, très, guère, extrêmement, réellement, plutôt, fort utile*

1.1. Utile ø

Ex (P) *le simulacre et la duperie mis en œuvre à bon escient se révèlent ici, plus que dans d'autres sports, d'**utiles** adjuvants.*

Σ UTILE Ø

⇆ INUTILE

(M) *Pour le reste, c'est-à-dire le travail, lorsqu'il est fastidieux, les robots sont **utiles**.*

∝ EFFICACE, NECESSAIRE, IMPORTANT, BON, MAUVAIS, NUISIBLE

1.2. Utile à, pour qqx

Ex (O) *L'évolution du conflit nécessita de mobiliser des recherches **utiles à** la Défense nationale.*

Σ UTILE À, POUR X_{QQX} [P.PERS] UTILE

(M) *je suis sûr qu'il y a là un investissement **utile pour** Paris et la France.*

(A) *Le père Joseph n'est pas marié avec sa vieille : ils ne sont qu ' « encabanés » . Elle **lui** est bien **utile**. Elle faut ses 40 kilomètres dans sa journée.*

1.3. Utile à, pour, de [V. inf]

Ex (M) *Je suis un responsable politique et je verrai ce qu'il y a de bon et d'**utile à** faire…*

Σ UTILE À, POUR, DE [V. INF]

(P) *Le savoir-faire des conseillers en communication s'avère alors particulièrement **utile pour** trouver les moyens de cette adaptation.*

(A) *je flaire la trahison dans le traître, et je trouve **utile de** dénoncer le criminel avant le crime.*

1.4. $Il_{[impersonnel]}$ [V] utile de [V. inf], que

Ex (M) ***Il est utile de** rappeler que le taux de mortalité y fut supérieur à celui des camps de concentration nazis.*

Σ $IL_{IMPERSONNEL}$ [V] UTILE DE [V. INF], QUE

(A) ***Il devenait utile de** se taire en effet, car il commençait à faire un peu jour.*

(J) *Il serait en conséquence extrêmement **utile que** l'honorable parlementaire puisse fournir des informations supplémentaires…*

▲	**V**	*Apparaître, paraître, sembler, utile de*

1.5. ≈ Utilisable

Ex (P) *Les performances sont les suivantes : accessibilité 600 mm ; vitesse maximum 10 cm/s, résolution 1/10 mm; charge **utile** 2 kg.*

Σ N UTILE

(M) *la Lyonnaise de banque, notamment, s'est fait sur la base du prix du mètre carré* ***utile*** *de 80 000 francs.* ± *+ postposé*

▲	**N**	*Charge utile*

1.6. Figements et semi-figements

1.6.1. ♦**A TOUTE(S) FIN(S) UTILE(S)**

1.6.2. ♦**FAIRE ŒUVRE UTILE**

1.6.3. ♦**EN TEMPS UTILE**

1.6.4. ♦**POUR UTILE QUE**
[non observé dans le corpus]

2. + Animé

Def Dont le travail, l'activité, les compétences sont ou peuvent être profitables aux autres, à la société. ⇐ L'UTILITÉ DE X_{QQN}

Σ N UTILE, UTILE N

Ex (A) *Tu es* ***utile****, mais Robespierre et Danton sont nécessaires.*

(M) *Avouer, c'est donner l'exemple et - - aussi et surtout - - se rendre* ***utile*** *pour l'avenir.* ∝ SAGE

(M) *les États mettent plus facilement à votre disposition des soldats que des policiers, qui sont pourtant extrêmement* ***utiles*** *dans des opérations de maintien de la paix.*

▼	**Sprep**	*Utile dans qqch.*

▲	**V[réfléchi]**	*Se rendre, se sentir utile*
	N	*Homme utile*

N.

Ex (A) *Il faudra sacrifier l'agréable à l'****utile****.*

C.9 Arrêter

ARRÊTER, *v.*

1. Arrêter

1.1. Arrêter qqn

⇐ L'ARRESTATION DE Y (PAR X)

Σ X_{QQN} ARRÊTE Y_{QQN}, Y_{QQN} EST ARRÊTÉ PAR X_{QQN}, Y_{QQN} SE FAIT ARRÊTER PAR X_{QQN}

± *+animé humain*

Def Appréhender quelqu'un et le retenir prisonnier.

Ex (M) *Le 21 mars, trois personnes sont ainsi **arrêtées** en flagrant délit dans une rue du douzième arrondissement de Paris…*

(M) *Trois mois plus tard, le 29 janvier, la police **arrête** plusieurs Thai à Anvers, dont Santi.*

∝ EMPRISONNÉ, INTERROGÉ, INCULPÉ

1.2. Arrêter qqn

⇐ Ø

Σ X_{QQX} ARRÊTE Y_{QQN}, RIEN [NEG] ARRÊTE Y_{QQN}

± *Avec négation le plus souvent*

Def Empêcher d'agir (par un obstacle).

Ex (O) *Rien ne l'**arrêtera** dans sa tendance pour devenir générale et philosophique.*

(A) *Aucune chaleur ne peut l'**arrêter**.*

▲ **Prep** *Rien ne l'arrêtera*

1.3. Arrêter qqx (quelque part)

1.3.1. Arrêter qqn (quelque part)

⇐ Ø

Σ X_{QQN} ARRÊTE Y_{QQN} (QUELQUE PART), X_{QQN} FAIT ARRÊTER Y_{QQN}, ÊTRE ARRÊTÉ (QUELQUE PART)

Def Empêcher d'avancer ou d'agir.

Ex (A) *tout le personnel de la Madeleine, étendant sur les marches du haut perron de cette église qui domine la rue Royale un large tapis rouge, faisait **arrêter** les passants.*

(A) *Sa Majesté lui arracha des mains le citron […] quand l'archevêque l'arrêtant lui dit à l'oreille : …*

▼ **Adv** *Arrêter net*

1.3.2. ⇐ L'arrêt de qqch.

⇐ L'ARRÊT DE Y (PAR X)

Σ X_{QQN} ARRÊTE Y_{QQCH}, X_{QQN} FAIT ARRÊTER Y_{QQCH}

Def Interrompre, stopper quelque chose.

Ex (M) *Alors nous avons **arrêté** le travail pour dire que cela devenait impossible de vivre comme cela.*

(M) *Un démenti ferme de ce groupe **arrêtait** le mouvement en fin de semaine.*

(A) *Cependant, ne voulant pas aller au café Anglais, il avait fait **arrêter** la voiture au coin de la rue.*

✻ *On n'arrête pas un train en marche*
[non observé dans le corpus]

1.3.3. Arrêter qqch.

⇐ Ø

Σ X_{QQN} ARRÊTE <*date…*>

Def Définir arbitrairement.

Ex (A) *On y passa des après-midi entiers à **arrêter** la forme d'une jupe.*

(M) *Lorsqu'il aura **arrêté** sa position, le gouvernement Balladur n'en sera pas quitte pour autant.*

▼	**(O)**	*Arrêter une date, un plan*

1.4. Arrêter de, pour [V. inf]

⇐ Ø

Σ X_{QQN} (S')ARRÊTE DE, POUR [V. INF]

Def Arrêter un processus, interrompre le déroulement de quelque chose.

Ex (M) *Carminatti **arrête de** se prendre pour Kirk Douglas dans Spartacus.*

(A) *Mais tout d'un coup il se mit à tousser, et s'**arrêta pour** laisser finir la quinte.*

(A) ***Arrêtez** mes amis ! **Arrêtez** !*
[sous-entendu arrêter de faire qqch., d'avancer]

1.5. Arrêter <*l'attention*>

⇐ Ø

Σ X_{QQN} ARRÊTE SON ATTENTION

Ex (2 occ) (A) *Par les considérations dont on environne souvent la présentation d'une hypothèse physique, il en est qui méritent d'**arrêter** notre **attention**.*

2. S'arrêter

2.1. S'arrêter Ø

2.1.1. S'arrêter

⇐ Ø

Σ X_{QQX} S'ARRÊTE
X_{QQX} S'ARRÊTE POUR [V. INF]

Def Cesser d'avancer, d'agir.

Ex (A) *La voiture venait de **s'arrêter**.*

(O) *Le mouvement **s'est arrêté** au début des années quatre-vingt, les expérimentations sociales ont alors été remisées au placard.*

∝ REPARTIR, REGARDER, SE RETOURNER

(A) *Ulenspiegel **s'arrêta**.*

(A) *Puis il entama la gorge, **s'arrêta** pour aiguiser son couteau émoussé, et arracha la tête du malheureux avant qu'elle ne fût coupée !*

▼	**N**	*S'arrêter un instant, un moment*
	Adv	*S'arrêter brusquement, tout à coup, encore, de nouveau, net*

2.1.1.1. ♦NE PAS S'ARRÊTER EN SI BON CHEMIN

2.1.2. <*temps*> qui s'arrête

⇐ L'ARRÊT DE X

Σ <*temps*> S'ARRÊTE

Ex (1 occ) (M) *Il envoya au ciel le service du Français. Roland-Garros suivit des yeux la chandelle, sa chute, alors que le temps **s'arrêtait**…*

2.1.3. ***<machine, objet>* qui s'arrête** ⇐ L'ARRÊT DE X

Σ < *machine, objet*> S'ARRÊTE

Def Cesser de fonctionner.

Ex (M) *Dans la nuit du 16 au 17 avril, entre Plymouth et Santander, un moteur* ***s'est arrêté*** *six heures durant, dans une mer démontée, mais aucun passager ne s'en est rendu compte.*

2.2. S'arrêter là

⇐ Ø

Σ X_{QQCH} S'ARRÊTE LÀ, LÀ S'ARRÊTE X_{QQCH}

Ex (M) *Mais* ***là*** *doit pour l'instant* ***s'arrêter*** *la comparaison.*

(M) *Ca* ***s'arrête là*** *(mais c'est déjà beaucoup), parce que, étrangement, dans ce diorama animé Laurent Pelly fait « dire », fait s'exprimer, ses acteurs, comme des manches.*

2.3. S'arrêter + prep

2.3.1. **S'arrêter, se faire arrêter quelque part s'y arrêter** ⇐ Ø

Σ X_{QQX} S'ARRÊTE, SE FAIT ARRÊTER QUELQUE PART
X_{QQX} S'Y ARRÊTE

Ex (A) *Un homme à cheval* ***s'arrêta*** *devant la chaumière.*

(A) *Il ressortit vers onze heures, erra quelque temps, prit un fiacre et* ***se fit arrêter*** *place de la Concorde.*

(M) *La ligne 4, qui* ***s'y arrête*** *en traversant Paris du nord au sud, étant maintenant étroitement surveillée.*

▼	**prep**	*S'arrêter devant, sur, au dessus, à, dans, sous, net qq part*

2.3.2. **S'arrêter à, sur qqch s'y arrêter** ⇐ Ø

Σ X_{QQX} S'ARRÊTE À, SUR X_{QQCH}
X_{QQX} S'Y ARRÊTE

Ex (O) *Le prosélytisme des convertis ne* ***s'arrête pas à*** *la littérature ou à la peinture.*

(A) *Son regard* ***s'arrêta sur*** *un enclos d'arbres.*

(P) *…l'analyse que McCloskey consacre à Fogel est […] la plus représentative de toute l'entreprise et mérite assurément qu'on* ***s'y arrête****.*

2.4. ♦ X S'ARRÊTE OÙ COMMENCE Y

⇐ Ø

Σ X_{QQCH} S'ARRÊTE OÙ COMMENCE Y_{QQCH}

Def Proverbe.

Ex (P) *La liberté de chacun* ***s'arrête où commence*** *celle d'autrui.*

(M) *La démocratie* ***s'arrête où commence*** *la volonté de l'état.*

N.

Ex (A) *des haies d'****arrête****-bœufs.*

(M) *L'****arrêté*** *signé…*

C.10 Comprendre

COMPRENDRE, *v.*

1. Comprendre

1.1. + Auxiliaire *avoir*

⇐ LA COMPRÉHENSION DE Y (PAR X)
X A COMPRIS Y

Σ X_{QQN} COMPREND Y_{QQX}

± *+aux. avoir*

Σ X_{QQX} COMPREND

Def Concevoir, saisir le sens de ; admettre.

1.1.1. (se faire) comprendre Ø

Ex (A) ***Comprends**-tu, grosse bedaine ?*

(M) ***Comprendre** pour prévenir au lieu de réprimer.*

(O) *Son amour pour la France est un absolu qu'un Français de race a du mal à **comprendre**.*
(M) *Comme s'il fallait **se faire** totalement **comprendre** […], pour juger sèchement le parti frère.*

▲	**V**	*Chercher à comprendre*
	N	*Difficulté à comprendre*
	Adj	*Difficile, facile à comprendre*

▼	**Neg**	*Ne pas comprendre*
	Adv	*Bien, mal comprendre*

1.1.2. (faire) comprendre que, ce que

Σ X_{QQN} COMPREND QUE, CE QUE

Ex (M) *J'ai alors **compris**, dit-elle aujourd'hui, **que** je ne serais jamais flic.*

(A) *Je n'ai pas **compris ce que** Jules Renard veut dire avec sa Vieille.*
(M) *[…] et **faire comprendre** […] que « la grève n ' est que la manifestation d ' un mal ».*

▲	**V**	*Chercher à comprendre que, ce que* *Permettre de comprendre que, ce que*

▼	**Neg**	*Ne pas comprendre que, ce que*
	Adv	*On comprend alors que* *Mieux comprendre que*

1.1.3. Faire comprendre qqch. à qqn
Faire comprendre à qqn que

Σ X_{QQN} FAIT COMPREND X_{QQCH} À Y_{QQN}

Ex (P) *Au départ, il était indispensable de **faire comprendre à** la population le fonctionnement d'une collectivité…*

(M) *Et Jacques Oudot de rappeler qu'il n'est pas aisé de **faire comprendre à** un élu **qu'**un rayon de lumière éphémère vaut 500 000 francs.*

1.1.4. (faire) comprendre qqx

Σ X_{QQN} (FAIT) COMPREND X_{QQX}

Ex (O) *Maintenant, le lecteur aura **compris** l'ambition de cet ouvrage, et le culot de ceux qui y ont travaillé.*

(A) ***Comprenez**-moi bien.*

(A) *Cet exemple suffira pour **faire comprendre** ma pensée.*

▲	**V**	*Chercher à comprendre qqch* *Essayer de comprendre qqch*
	Conj	*Comment comprendre qqch*
	Synt	*Et l'on comprend…*

▼		
	Neg	*Ne pas comprendre qqch*
	Adv	*Bien, mal comprendre qqch*
	Sprep	*Être mal, bien compris de qqn*

1.1.4.1. Comprenez + N

Ex (M) *« Une charmante » -- **comprenez** miniature, puisqu'elle ne mesure que quinze centimètres sur vingt --*

1.1.5. Comprendre + Subordonnée

Σ X_{QQN} COMPREND [SUBORDONNÉE]

Ex (P) *Il reste à **comprendre** pourquoi le déficit en vitamine E est responsable de troubles neurodégénératifs aussi spécifiques.*

(P) *Il faut donc chercher à **comprendre** à quoi renvoie la formule du langage courant.*

▼		
	Conj	*Comprendre pourquoi, comment, comme, quand, de quelle manière, d'où, à quoi, de quoi, quel*

1.1.6. Ne rien comprendre à

Σ NE RIEN COMPRENDRE À
N'Y RIEN COMPRENDRE

Ex (A) *Il **ne comprends rien à** la vie et se passionne à la regarder.*

(A) *Je **n'y comprends rien** !*

1.2. + Auxiliaire *être*

⇐ Ø
Y EST COMPRIS DANS X

Σ X_{QQX} COMPREND Y_{QQX}

± *+aux. être*

Def Inclure, englober ; être constitué de.

Ex (M) *Le conseil de la politique monétaire, outre le gouverneur et les deux sous-gouverneurs de la Banque de France, **comprend** six membres choisis en fonction de leurs compétences dans le domaine économique et monétaire.*

(O) *Elle **comprendrait** la mise en service des locomotives purement SNCF…*

▼		
	neg	*Non compris*

1.2.1. Être compris entre qqch et qqch

Σ X EST COMPRIS ENTRE Y ET Z

X_{QQCH} EST COMPRIS entre *<chiffre, date, lieu>* et *<chiffre, date, lieu>*

Ex (A) *il doit explorer tout le pays **compris entre** le Nil **et** le Tchad.*

(M) *Pour reconstruire un pays comme le Koweït, somme toute minuscule et peu peuplé, certains évoquent des montants **compris entre** 50 **et** 100 milliards de dollars !*

1.2.2. N compris(e)

Σ N COMPRIS(E)

Ex (O) *[…] chauffage **compris**.*

(J) *[…] l'Italie **comprise**.*

1.3. ♦Y COMPRIS, NON COMPRIS

Σ Y COMPRIS
[en général précédé de la virgule]

Ex (M) *Cette mission, elle la remplit actuellement, **y compris** jusqu'à Sarajevo.*

(M) *Toute pression, **y compris** celles qui viendraient d'administrations publiques étrangères, à l'occasion de ventes, doivent être signalées.*

2. Se comprendre

2.1. + Animé

Σ X_{QQN} ET Y_{QQN} SE COMPRENNE
X_{QQN} SE COMPREND AVEC Y_{QQN}

Def S'entendre, être en harmonie avec quelqu'un.

Ex (A) *Leurs yeux se rencontraient, semblaient se concerter,* ***se comprendre****…*

2.2. - Animé

X_{QQCH} SE COMPREND

Def Admettre, interpréter quelque chose.

Ex (O) *Même si la solution apportée par la cour de Paris* ***se comprend*** *du point vue de l'équité…*

(P) *N'oublions pas que l'argumentation devrait avant tout* ***se comprendre*** *dans un contexte bien particulier…*

N.

Ex (O) *Parlant de Dieu, du catholicisme, de sa foi, de son bonheur, et comme je lui disais* ***le*** *bien* ***comprendre****, il ajouta…*

R.

Ex (P) *FAGES Jean – Baptiste,* ***Comprendre*** *René Girard, Toulouse, Privat, 1982.*

C.11 Importer

IMPORTER, *v.*

1. ⇐ Importance

1.1. Importer (à, pour qqx)

⇐ X A DE L'IMPORTANCE POUR Y
X EST IMPORTANT POUR Y

Σ X_{QQCH} IMPORTE Ø
X_{QQCH} IMPORTE À, POUR Y_{QQCH}

(A) *Ce qui **importe**, c'est le chef.*

Ex (O) *[…] sont-ils bien assurés que les races […] ne possèdent point de facultés et des aptitudes dont la conservation et le développement ultérieur **importent à** l'avenir de l'espèce.*

(P) *Il écrit que « Ce qui **m'importe** est la réintégration dans et contre les Lumières…*

(P) *Si on fait le bilan des contrôles que les partis et les médias ont sur les différentes ressources pouvant leur servir d'atouts dans les actions qui **importent pour** eux…*

▼	**Adv**	*Importer peu, moins*

▲	**Pro (S)**	*Ce qui importe c'est*
	Adj (S)	*Seul x importe*

1.2. Qu'/peu importe qqch. (à qqn)
Qu'/Peu importe (à qqn) que

⇐ X A PEU D'IMPORTANCE POUR Y

Σ QU'IMPORTE X_{QQCH} (À Y_{QQN})
PEU IMPORTE X_{QQCH} (À Y_{QQN})
QU'IMPORTE (À Y_{QQN}) QUE
PEU IMPORTE (À Y_{QQN}) QUE

Ex (A) ***Qu'importaient** de pareilles misères !*

(A) ***Que m'importe** la tempête, si j'ai la boussole…*

(A) ***Peu importait à** Bouvard le procédé.*

(M) ***Peu importe à** Me Vergès l'arrêt de la Cour de cassation…*

(O) ***Peu importe** à cet égard **qu'**une idée soit originale ou non.*

1.3. $Il_{impersonnel}$ importe (à, pour qqn) de [V. inf]
$Il_{impersonnel}$ importe (à, pour qqn) que

⇐ IL EST IMPORTANT DE, QUE

Σ $IL_{IMPERSONNEL}$ IMPORTE (À, POUR X_{QQN}) DE, QUE

Ex (O) ***Il importe** bien **de** savoir si nous faisons de la sociologie.*

(O) ***Il nous importe** donc **de** montrer et d'expliquer à ces patrons ce qui marche bien…*

(A) ***Il importe que** personne encore ne sache mon nom.*

(A) ***Il lui importe** peu **que** la cosmologie atomiste s'en déclare satisfaite ou que les principes du Cartésianisme le condamnent.*

▼	**Adv**	*Il importe peu de*

▲	**Pro (S)**	*Ce qu'il importe c'est de*

1.4. N'/peu importe [adv, pro, conj]

⇐ Ø

Σ N'/PEU IMPORTE <*lequel, laquelle, qui, quoi, où, quand, comment, si…*>

Ex (M) *Il ne faut pas y emmener **n'importe qui**.*

(M) ***Peu importe où** l'on frappe tout au long d'un front qui s'étend sur des centaines de kilomètres.*

1.5. Qu'/peu/n'importe Ø

⇐ ÇA N'A PAS D'IMPORTANCE

Σ QU'IMPORTE
PEU IMPORTE
N'IMPORTE

± ±*point d'exclamation*

Ex (A) *Elle ou un autre,* ***qu'importait*** *!*

(M) *Oui, ici les acteurs et les autres font tout sur la scène, mais la société, bon,* ***peu importe****, ce n'est pas ici un discours de François Mitterrand.*

(A) *La terre buvait l'eau,* ***n'importe*** *!*

2. ⇐ Importation

⇐ L'IMPORTATION DE Y PAR X QUELQUE PART

L'IMPORTATION DE Y PAR X DE QUELQUE PART

⇆ EXPORTER

2.1. Importer <*objets, marchandises*>

Σ X_{QQN} IMPORTE <*objet, marchandise*> DANS, VERS <*pays, région*>

Ex (M) *Comment les industries japonaises espèrent-elles survivre si nous sommes contraints d'****importer*** *l'acier pour construire nos navires et nos ponts ?*

2.2. Importer <*population, main d'œuvre*>

Σ X_{QQN} IMPORTE <*population, main d'oeuvre*> DANS, VERS <*pays, région*>

Ex (O) *On peut faire venir quelques spécialistes de n'importe où, on ne peut pas* ***importer*** *toute la population.*

2.3. Importer <*concepts, idées*>

Σ X_{QQN} IMPORTE <*concept, idée*> DANS, VERS <*culture, discipline*>

Ex (O) *Des catégories sociologiques ou des concepts politiques issus des sciences sociales sont* ***importées*** *dans les sciences naturelles.*

(M) *C'est l'américain Dell qui a* ***importé*** *cette méthode en France.*

Annexe D

Fréquence des lexies

Le tableau D.1 ci-dessous donne pour chacune des lexies de chacun de nos vocables la fréquence de ses occurrences dans le corpus et dans chacun des sous-corpus.

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
arrêter	1.1	22	15	55	4	0	96
arrêter	1.2	4	0	1	1	0	6
arrêter	1.3.1	62	0	4	0	0	66
arrêter	1.3.2	43	23	29	14	27	136
arrêter	1.3.3	11	160	21	4	0	196
arrêter	1.4	17	0	9	0	4	30
arrêter	1.5	1	0	0	0	1	2
arrêter	2.1.1	175	1	19	12	12	219
arrêter	2.1.1.1	0	0	0	1	0	1
arrêter	2.1.2	0	0	1	0	0	1
arrêter	2.1.3	0	0	1	0	0	1
arrêter	2.2	3	2	10	9	2	26
arrêter	2.3.1	71	2	8	5	1	87
arrêter	2.3.2	16	1	3	9	16	45
arrêter	2.4	0	0	2	0	2	4
barrage	1	0	1	0	0	0	1
barrage	2.1.1	1	57	4	4	4	70
barrage	2.1.2	0	0	12	0	0	12
barrage	2.2.2	0	0	1	4	0	5
barrage	2.2.2.1	0	1	3	0	0	4
biologique	1	0	33	11	305	78	427
biologique	1.1	0	0	0	0	1	1
biologique	2	0	6	1	0	0	7
biologique	3	0	36	3	1	0	40
chef	1.1	154	80	386	165	76	861
chef	1.1.1	5	1	15	2	0	23
chef	1.2	11	5	28	18	7	69
chef	1.3	0	0	15	1	1	17
chef	2	0	5	3	15	0	23
chef	3.1	3	1	0	0	1	5
chef	3.2	21	0	1	0	0	22
chef	4	5	1	4	1	1	12
chef	5.1	37	1	25	7	6	76
chef	5.2	0	5	5	5	8	23

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
chef	5.3	0	0	1	1	0	2
clair	1.1	39	0	0	0	1	40
clair	1.2	0	0	1	0	0	1
clair	1.3	34	0	6	0	1	41
clair	1.4.1	19	0	2	0	0	21
clair	1.4.2	27	0	1	0	1	29
clair	1.5	2	0	0	0	0	2
clair	1.6	8	0	0	0	0	8
clair	1.7	1	0	0	0	0	1
clair	2.1	38	13	36	39	37	163
clair	2.2	9	12	16	9	16	62
clair	2.3	12	18	15	41	38	124
clair	2.4	1	0	0	0	0	1
clair	2.5	0	0	1	0	0	1
clair	3	15	0	2	0	1	18
clair	4.1	0	0	1	0	0	1
clair	4.2	1	0	5	0	0	6
clair	4.3	0	0	4	1	0	5
clair	4.4	11	0	4	5	2	22
clair	4.5	0	1	7	0	2	10
clair	4.6	0	0	1	0	0	1
communication	1.1	2	45	37	30	577	691
communication	1.1.1	0	0	14	0	14	28
communication	1.1.2	0	0	0	0	393	393
communication	1.1.3	0	0	0	0	22	22
communication	1.2	6	7	1	10	19	43
communication	1.3.1	0	0	0	0	8	8
communication	1.3.2	0	0	1	0	10	11
communication	1.4	7	20	5	1	12	45
communication	2.1	0	5	0	0	1	6
communication	2.2	2	260	10	23	20	315
communication	2.3	0	40	0	12	11	63
communication	3	3	19	4	3	37	66
communication	4	0	1	2	1	8	12
compagnie	1.1	15	121	84	59	15	294
compagnie	1.2	11	1	2	1	0	15
compagnie	1.3	1	0	12	3	1	17
compagnie	1.4	1	0	0	0	0	1
compagnie	2	4	0	0	0	2	6
compagnie	2.1	18	2	20	3	6	49
compagnie	2.2	5	6	0	0	0	11
compagnie	2.2.1	2	0	3	0	0	5
compagnie	2.2.2	1	0	0	0	0	1
compagnie	2.3	7	0	0	0	0	7
compagnie	2.4	5	0	0	0	0	5
compagnie	2.5	0	0	0	1	0	1
comprendre	1.1.1	121	2	24	12	32	191
comprendre	1.1.2	67	7	77	71	62	284
comprendre	1.1.3	12	3	10	6	5	36
comprendre	1.1.4	156	19	74	173	278	700
comprendre	1.1.4.1	0	0	2	0	0	2
comprendre	1.1.5	8	2	13	39	49	111

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
comprendre	1.1.6	23	0	5	1	1	30
comprendre	1.2	15	141	32	63	61	312
comprendre	1.2.1	6	12	9	1	21	49
comprendre	1.2.2	1	4	7	4	3	19
comprendre	1.3	4	202	66	54	53	379
comprendre	2.1	2	0	2	1	3	8
comprendre	2.2	2	1	0	10	11	24
concentration	1	0	2	8	0	1	11
concentration	2.1	0	55	12	5	2	74
concentration	2.2	0	0	4	4	4	12
concentration	2.3	0	0	8	10	13	31
concentration	3	0	51	1	2	57	111
concentration	4	0	0	3	0	4	7
conclure	1.1	19	4	34	26	29	112
conclure	1.1.1	0	0	2	0	1	3
conclure	1.2.1	0	0	4	20	2	26
conclure	1.2.2	0	3	3	0	0	6
conclure	1.2.3	8	238	45	38	2	331
conclure	1.3.1	0	16	6	16	17	55
conclure	1.3.1.1	0	0	0	0	4	4
conclure	1.3.2	0	3	2	3	2	10
conclure	1.3.3	1	0	1	2	2	6
conclure	1.4.1	2	0	2	2	8	14
conclure	1.4.2	22	42	11	25	49	149
conclure	1.5	1	0	0	0	0	1
conclure	1.6	0	0	1	0	0	1
conclure	2.1	1	0	1	0	1	3
conclure	2.2	0	0	1	3	1	5
conclure	2.3	0	0	0	0	1	1
conduire	1.1	10	24	30	18	40	122
conduire	1.2.1	22	65	50	103	177	417
conduire	1.2.2	15	29	50	134	128	356
conduire	1.3.1	6	0	4	0	1	11
conduire	1.3.2	7	1	3	7	1	19
conduire	1.3.3	0	1	6	0	0	7
conduire	1.4.1	60	2	23	14	8	107
conduire	1.4.2	4	0	9	8	5	26
conduire	1.4.3	7	0	0	0	0	7
conduire	1.4.3.1	1	0	0	0	0	1
conduire	1.5	0	0	1	0	0	1
conduire	1.6.1	1	0	0	0	0	1
conduire	1.6.2	4	0	0	0	0	4
conduire	2.1	7	0	2	3	1	13
conduire	2.2	0	0	0	1	0	1
connaître	1.1	225	102	65	128	135	655
connaître	1.1.1	0	0	0	1	0	1
connaître	1.2	14	3	7	7	5	36
connaître	1.2.1	4	0	3	1	2	10
connaître	1.2.2	1	0	0	0	0	1
connaître	1.2.3	0	2	0	9	0	11
connaître	1.3	17	89	89	174	91	460
connaître	1.4	5	0	2	0	1	8

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
connaître	1.5	181	0	36	41	12	270
connaître	1.6	35	44	8	21	25	133
connaître	1.7	2	0	1	0	0	3
connaître	1.8.1	1	0	0	1	0	2
connaître	1.8.2	1	0	0	0	0	1
connaître	2.1	17	1	4	7	4	33
connaître	2.2	2	0	1	0	0	3
connaître	2.3	6	0	1	1	0	8
constitution	1	4	17	89	23	16	149
constitution	1.1	1	0	1	0	0	2
constitution	2	6	37	14	89	65	211
constitution	3	12	0	0	4	5	21
constitution	4	3	0	0	1	0	4
constitution	5	0	0	4	31	0	35
correct	1.1	17	0	0	1	1	19
correct	1.2	3	25	4	9	21	62
correct	1.3	0	0	1	1	2	4
correct	1.4	0	3	2	2	1	8
correct	2	0	6	4	5	8	23
courant	1.1	4	25	20	54	50	153
courant	1.1.1	0	0	2	1	3	6
courant	1.2	0	3	1	3	0	7
courant	2	2	1	1	0	0	4
couvrir	1.1	0	6	6	1	4	17
couvrir	1.10	0	0	1	0	0	1
couvrir	1.2	0	0	0	3	2	5
couvrir	1.3.1	44	5	9	2	9	69
couvrir	1.3.2	14	0	1	0	1	16
couvrir	1.3.3	10	0	0	0	0	10
couvrir	1.4	1	154	4	11	11	181
couvrir	1.4.1	0	2	1	1	6	10
couvrir	1.5	0	27	2	3	4	36
couvrir	1.6	0	69	12	6	3	90
couvrir	1.7.1	1	0	1	0	0	2
couvrir	1.7.2	1	7	7	1	4	20
couvrir	1.7.3	0	16	2	8	2	28
couvrir	1.8	2	0	3	1	1	7
couvrir	1.9.1	15	0	1	2	0	18
couvrir	1.9.2	4	0	0	2	0	6
couvrir	1.9.3	2	0	0	0	0	2
couvrir	2.1	3	0	0	0	0	3
couvrir	2.1.1	3	0	0	0	0	3
couvrir	2.2.1	6	0	0	0	0	6
couvrir	2.2.2	6	0	2	0	0	8
couvrir	2.3	0	0	2	2	1	5
degré	1.1	34	1	3	3	7	48
degré	1.2	26	0	0	0	4	30
degré	1.3	4	5	0	0	1	10
degré	1.4	0	0	0	0	1	1
degré	2	9	0	0	0	0	9
degré	3.1	0	0	2	4	3	9
degré	3.2	38	46	11	77	125	297

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
degré	3.3	6	1	2	4	7	20
degré	3.4	0	3	3	6	0	12
degré	4	1	0	1	0	2	4
degré	5.1	0	0	0	1	0	1
degré	5.10	0	0	0	0	4	4
degré	5.2	0	4	1	1	6	12
degré	5.3	0	1	3	2	1	7
degré	5.4	9	0	2	2	2	15
degré	5.5	0	0	1	1	11	13
degré	5.7	0	0	10	0	0	10
degré	5.9	1	0	0	4	0	5
détention	1	1	42	34	4	0	81
détention	2	0	23	5	2	1	31
entrer	1.1	160	1	6	1	3	171
entrer	1.2	271	5	27	15	17	335
entrer	1.2.1	20	7	1	1	0	29
entrer	1.2.2	14	0	0	0	1	15
entrer	1.2.3	3	0	2	7	3	15
entrer	1.3.1	5	0	0	1	0	6
entrer	1.3.2	3	0	0	0	0	3
entrer	1.3.3	7	0	1	2	0	10
entrer	1.4	18	4	26	46	9	103
entrer	1.4.1	2	0	2	2	0	6
entrer	1.4.12	2	0	0	0	0	2
entrer	1.4.2	0	0	1	0	0	1
entrer	1.4.3	0	0	1	0	0	1
entrer	1.4.4	2	0	3	3	7	15
entrer	1.4.6	0	0	1	0	1	2
entrer	1.5.1	11	55	13	44	28	151
entrer	1.5.2	1	0	6	0	2	9
entrer	1.5.3	2	0	2	1	5	10
entrer	1.6	1	0	1	0	2	4
entrer	1.6.1	5	0	0	3	0	8
entrer	1.6.2	3	9	21	9	23	65
entrer	1.6.2.1	0	0	3	0	0	3
entrer	1.6.2.2	0	0	4	0	0	4
entrer	1.6.2.3	0	1	0	0	0	1
entrer	1.6.3	2	0	1	5	0	8
entrer	1.7.1	14	0	1	0	0	15
entrer	1.7.2	0	3	1	18	18	40
entrer	1.7.3	2	15	6	9	2	34
entrer	1.7.4	0	116	12	4	1	133
entrer	1.7.5	0	0	2	2	1	5
entrer	1.7.6	2	1	0	1	1	5
entrer	1.7.7	1	2	2	3	3	11
entrer	1.7.8.1	3	1	1	3	5	13
entrer	1.7.8.2	0	1	0	0	0	1
entrer	1.7.8.3	1	0	4	0	0	5
entrer	1.7.8.4	0	6	0	1	4	11
entrer	1.7.8.5	0	0	0	0	3	3
entrer	1.7.8.6	0	0	1	0	0	1
entrer	2	0	0	1	1	2	4

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
exceptionnel	1	3	14	41	35	27	120
exceptionnel	2	2	12	14	18	19	65
exceptionnel	3	0	23	7	10	1	41
exercer	1.1	0	93	32	22	13	160
exercer	1.2.1	41	102	46	139	87	415
exercer	1.2.2	1	0	2	0	2	5
exercer	1.3	1	0	0	0	1	2
exercer	2.1	0	0	0	2	1	3
exercer	2.2	18	17	11	20	41	107
exercer	2.3	3	0	0	0	0	3
exercer	2.3.1	2	0	1	0	0	3
formation	1.1	2	460	128	325	60	975
formation	1.2.1	0	160	14	22	3	199
formation	1.2.2	0	7	12	8	2	29
formation	1.2.3	0	1	1	2	2	6
formation	1.2.4	0	0	1	1	0	2
formation	1.3	0	0	1	12	3	16
formation	2.1	10	17	11	41	148	227
formation	2.2	1	4	34	6	20	65
formation	3	1	2	1	2	3	9
frais	1.1	21	0	1	0	2	24
frais	1.1.1	1	0	0	0	0	1
frais	1.2	15	0	1	0	1	17
frais	1.2.1	2	0	0	0	0	2
frais	2	5	0	0	0	0	5
frais	3.1	7	0	0	1	1	9
frais	3.1.1	0	0	1	0	0	1
frais	3.1.2	1	1	0	3	0	5
frais	3.1.3	3	0	3	0	0	6
frais	3.1.4	0	0	2	0	0	2
frais	3.2.1	1	0	0	0	0	1
frais	3.2.2	3	1	0	0	0	4
frais	3.2.3	22	0	0	0	0	22
frais	3.3	10	0	0	1	0	11
frais	4.1	3	0	0	0	0	3
frais	4.2	2	0	0	0	0	2
frais	5	11	53	2	0	1	67
frais	6	2	0	0	0	0	2
haut	1.1	116	1	13	4	0	134
haut	1.2	9	0	2	0	1	12
haut	1.3	4	2	3	0	1	10
haut	1.4	6	3	4	0	11	24
haut	1.4.1	5	18	18	3	8	52
haut	1.4.2	9	0	2	0	0	11
haut	1.4.2.1	1	0	0	0	0	1
haut	1.5	14	0	0	0	0	14
haut	1.6	5	0	0	0	0	5
haut	1.6.1	7	0	0	1	0	8
haut	1.6.2	3	0	0	1	0	4
haut	1.7	8	0	1	0	4	13
haut	10.1	1	0	2	0	1	4
haut	10.2	0	0	2	2	1	5

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
haut	10.3	0	0	1	0	0	1
haut	10.4	1	0	2	0	0	3
haut	10.5	2	0	0	0	0	2
haut	10.6	0	0	3	0	0	3
haut	10.7	2	0	0	0	0	2
haut	10.8	0	1	0	0	0	1
haut	2	0	0	2	3	0	5
haut	2.1	8	3	1	0	0	12
haut	3	26	0	3	0	0	29
haut	4	2	14	6	18	80	120
haut	5	1	1	7	2	2	13
haut	6	24	44	67	24	28	187
haut	7	18	65	74	47	50	254
haut	8	12	1	6	8	3	30
haut	9	3	27	10	7	11	58
historique	1.1	16	74	83	122	247	542
historique	1.2	0	3	23	14	8	48
historique	1.3	1	4	1	10	14	30
importer	1.1	11	0	5	15	24	55
importer	1.2	19	0	9	10	11	49
importer	1.3	8	40	6	44	28	126
importer	1.4	26	9	35	30	59	159
importer	1.5	37	0	8	1	3	49
importer	2.1	1	92	10	16	0	119
importer	2.2	0	0	0	1	1	2
importer	2.3	1	0	3	8	5	17
lancement	1	0	8	5	1	5	19
lancement	1.1	0	0	5	0	1	6
lancement	2	0	43	10	46	11	110
lancement	2.1	0	0	2	0	0	2
lancement	3	0	0	0	0	1	1
mettre	1.1.1	294	1	28	22	7	352
mettre	1.1.10.1	0	1	0	4	1	6
mettre	1.1.10.2	2	0	0	0	0	2
mettre	1.1.11	0	0	2	0	0	2
mettre	1.1.12.1	0	0	1	0	0	1
mettre	1.1.12.10	0	0	0	0	1	1
mettre	1.1.12.11	0	0	1	0	0	1
mettre	1.1.12.12	1	0	0	0	0	1
mettre	1.1.12.13	0	0	4	0	0	4
mettre	1.1.12.14	0	0	1	0	0	1
mettre	1.1.12.15	0	0	1	0	0	1
mettre	1.1.12.16	0	0	2	0	0	2
mettre	1.1.12.17	2	0	2	5	0	9
mettre	1.1.12.18	0	0	1	0	0	1
mettre	1.1.12.19	0	0	1	0	1	2
mettre	1.1.12.20	1	0	1	0	0	2
mettre	1.1.12.21	2	2	1	2	0	7
mettre	1.1.12.22	0	0	1	0	0	1
mettre	1.1.12.23	0	0	1	0	0	1
mettre	1.1.12.24	0	1	0	0	0	1
mettre	1.1.12.25	0	1	0	0	0	1

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
mettre	1.1.12.26	0	1	0	0	0	1
mettre	1.1.12.3	0	0	2	2	3	7
mettre	1.1.12.4	1	0	0	0	0	1
mettre	1.1.12.5	4	0	4	0	0	8
mettre	1.1.12.9	1	0	0	0	0	1
mettre	1.1.2	22	0	1	0	0	23
mettre	1.1.2.1	0	0	1	0	0	1
mettre	1.1.3	9	0	2	2	1	14
mettre	1.1.3.1	0	0	1	0	0	1
mettre	1.1.3.2	0	0	1	0	0	1
mettre	1.1.3.3	0	1	0	0	0	1
mettre	1.1.4	5	0	0	0	0	5
mettre	1.1.5	6	1	2	2	0	11
mettre	1.1.5.1	3	0	0	0	0	3
mettre	1.1.6	12	1	9	7	4	33
mettre	1.1.7	39	0	6	7	2	54
mettre	1.1.8	38	3	7	2	2	52
mettre	1.1.9	9	0	3	1	2	15
mettre	1.10	4	0	2	0	1	7
mettre	1.11	1	0	0	0	0	1
mettre	1.12.1	0	0	3	0	2	5
mettre	1.12.10	0	0	4	0	0	4
mettre	1.12.11	0	0	0	2	1	3
mettre	1.12.12	6	0	0	1	0	7
mettre	1.12.13	1	0	0	0	0	1
mettre	1.12.14	0	2	3	2	4	11
mettre	1.12.15	5	1	1	1	0	8
mettre	1.12.16	4	2	1	1	0	8
mettre	1.12.17	5	0	3	0	1	9
mettre	1.12.18	1	0	2	1	0	4
mettre	1.12.19	0	0	1	0	0	1
mettre	1.12.2	4	0	3	0	0	7
mettre	1.12.20	2	1	3	3	2	11
mettre	1.12.21	0	0	1	0	0	1
mettre	1.12.22	0	0	1	0	0	1
mettre	1.12.23	0	0	0	0	1	1
mettre	1.12.24	0	0	1	0	1	2
mettre	1.12.25	1	0	2	0	0	3
mettre	1.12.26	0	0	1	0	0	1
mettre	1.12.27	1	0	1	0	1	3
mettre	1.12.28	0	0	0	1	0	1
mettre	1.12.29	0	0	1	0	0	1
mettre	1.12.3	4	26	15	2	6	53
mettre	1.12.30	0	0	0	2	1	3
mettre	1.12.31	1	1	2	1	0	5
mettre	1.12.32	0	0	2	0	0	2
mettre	1.12.33	1	0	0	0	0	1
mettre	1.12.34	0	0	0	0	1	1
mettre	1.12.35	0	1	0	0	0	1
mettre	1.12.36	1	0	0	1	0	2
mettre	1.12.4	0	28	17	19	41	105
mettre	1.12.5	6	0	0	0	0	6

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
mettre	1.12.6	1	41	28	20	108	198
mettre	1.12.7	4	126	58	33	9	230
mettre	1.12.8	4	0	3	2	2	11
mettre	1.12.9	0	0	3	0	0	3
mettre	1.2	189	473	375	520	659	2216
mettre	1.2.1.1	5	384	58	86	107	640
mettre	1.2.1.10	3	26	2	16	22	69
mettre	1.2.1.11	4	0	0	1	0	5
mettre	1.2.1.12	0	0	3	0	0	3
mettre	1.2.1.13	1	0	2	1	3	7
mettre	1.2.1.14	0	0	1	0	0	1
mettre	1.2.1.15	0	0	1	0	0	1
mettre	1.2.1.16	0	0	1	0	0	1
mettre	1.2.1.17	1	1	0	0	0	2
mettre	1.2.1.18	0	2	0	0	0	2
mettre	1.2.1.2	4	0	0	1	2	7
mettre	1.2.1.3	2	0	0	0	0	2
mettre	1.2.1.4	0	3	0	2	4	9
mettre	1.2.1.5	2	11	7	21	17	58
mettre	1.2.1.6	0	0	2	0	0	2
mettre	1.2.1.7	0	0	2	0	1	3
mettre	1.2.1.8	1	0	0	1	0	2
mettre	1.2.1.9	1	3	4	5	7	20
mettre	1.2.12	0	0	1	0	1	2
mettre	1.3.1	36	5	10	42	4	97
mettre	1.3.2	7	0	0	0	0	7
mettre	1.4	61	0	1	0	0	62
mettre	1.4.1	2	0	0	0	0	2
mettre	1.5	12	0	14	7	4	37
mettre	1.6.1	15	0	4	5	1	25
mettre	1.6.1.1	0	0	0	1	0	1
mettre	1.6.2	4	0	0	4	0	8
mettre	1.7	13	1	5	2	1	22
mettre	1.8	0	27	2	2	0	31
mettre	1.9	0	0	0	0	1	1
mettre	2.1	30	0	3	0	1	34
mettre	2.2	16	0	0	2	0	18
mettre	2.3.1	13	0	2	0	0	15
mettre	2.3.2	243	1	23	15	15	297
mettre	2.3.3	5	0	5	2	0	12
mettre	2.3.4	2	0	0	0	0	2
mettre	2.4	50	12	26	10	7	105
mettre	2.5	4	0	0	0	0	4
mettre	2.6	1	0	1	0	0	2
mettre	2.7	1	0	0	0	0	1
mettre	2.7.1	2	0	0	0	0	2
mettre	2.8.1	9	0	0	0	0	9
mettre	2.8.10	1	0	0	2	0	3
mettre	2.8.11	1	0	0	0	0	1
mettre	2.8.12	1	0	2	0	4	7
mettre	2.8.13	0	0	0	0	1	1
mettre	2.8.14	0	0	0	1	0	1

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
mettre	2.8.15	0	0	1	0	0	1
mettre	2.8.16	0	0	0	0	1	1
mettre	2.8.17	3	0	0	0	0	3
mettre	2.8.18	0	0	1	0	0	1
mettre	2.8.19	0	0	0	1	1	2
mettre	2.8.2	12	0	0	0	0	12
mettre	2.8.20	0	0	1	0	0	1
mettre	2.8.21	0	0	1	0	0	1
mettre	2.8.3	0	1	1	0	0	2
mettre	2.8.4	1	0	0	0	0	1
mettre	2.8.5	0	1	2	0	0	3
mettre	2.8.6	2	0	0	0	0	2
mettre	2.8.7	0	0	0	1	0	1
mettre	2.8.8	0	0	2	0	0	2
mettre	2.8.9	0	0	1	0	0	1
observation	1	113	24	25	116	214	492
observation	2	9	45	3	2	7	66
observation	3	0	8	0	6	0	14
organe	1.1	0	4	1	18	8	31
organe	1.2	2	1	7	0	2	12
organe	1.3	0	1	0	44	0	45
organe	1.4	1	55	14	17	7	94
organe	2	28	16	7	76	13	140
organe	3	0	0	0	44	0	44
ouvrir	1.1.1	20	0	1	0	1	22
ouvrir	1.1.2	5	0	0	0	0	5
ouvrir	1.2.1	206	0	11	10	12	239
ouvrir	1.2.1.1	3	0	5	0	0	8
ouvrir	1.2.1.2	6	0	1	0	0	7
ouvrir	1.2.1.3	3	0	4	2	0	9
ouvrir	1.2.2	2	14	17	1	2	36
ouvrir	1.2.2.1	0	0	8	1	0	9
ouvrir	1.2.3	6	0	0	0	0	6
ouvrir	1.2.4	1	1	2	0	1	5
ouvrir	1.3	11	20	36	26	79	172
ouvrir	1.3.1	0	6	4	10	27	47
ouvrir	1.3.2	0	3	1	7	1	12
ouvrir	1.4.1	2	11	28	8	13	62
ouvrir	1.4.1.1	1	0	1	0	0	2
ouvrir	1.4.1.2	0	2	4	0	0	6
ouvrir	1.4.1.3	1	0	1	0	0	2
ouvrir	1.4.2	0	45	7	7	3	62
ouvrir	1.5	1	0	0	0	0	1
ouvrir	1.6	4	0	2	0	1	7
ouvrir	1.7	0	0	1	0	0	1
ouvrir	1.8.1	3	1	0	3	2	9
ouvrir	1.8.2	1	0	0	0	0	1
ouvrir	1.9.1	0	0	1	0	0	1
ouvrir	1.9.2	0	0	0	0	1	1
ouvrir	1.9.3	1	0	0	0	0	1
ouvrir	2.1.1	1	2	13	5	4	25
ouvrir	2.1.2	2	1	3	1	1	8

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
ouvrir	2.2	0	0	2	0	1	3
ouvrir	2.3	1	0	0	0	0	1
ouvrir	2.3.1	53	0	6	1	6	66
ouvrir	2.3.2	7	0	0	0	0	7
ouvrir	2.4.1	3	0	0	0	0	3
ouvrir	2.4.2	2	0	0	0	0	2
ouvrir	2.5.1	1	1	1	3	9	15
ouvrir	2.5.2	0	3	9	9	9	30
ouvrir	2.6	1	0	0	1	0	2
ouvrir	2.7.1	13	0	2	0	0	15
ouvrir	2.7.2	6	0	0	0	1	7
ouvrir	2.8	0	0	0	0	1	1
ouvrir	2.9	0	0	0	0	1	1
parvenir	1.1.1	10	58	26	52	15	161
parvenir	1.1.2	1	1	2	5	1	10
parvenir	1.2	43	13	61	59	64	240
parvenir	1.2.1	2	5	6	2	3	18
parvenir	1.3	12	4	18	18	13	65
parvenir	1.4	47	36	9	9	14	115
parvenir	2	2	29	8	4	1	44
parvenir	3	1	0	0	0	0	1
passage	1	26	0	7	2	2	37
passage	10.1	0	0	2	0	0	2
passage	10.2	0	0	1	0	0	1
passage	2	27	2	8	31	33	101
passage	3.1	31	24	27	17	27	126
passage	3.2	1	0	1	0	1	3
passage	4.1	0	2	1	6	8	17
passage	4.1.1	1	0	0	0	0	1
passage	4.2	0	0	0	0	6	6
passage	5	3	0	3	2	2	10
passage	6	10	12	12	102	86	222
passage	7	0	1	0	1	0	2
passage	8	0	4	1	3	4	12
passage	9.1	8	1	17	5	10	41
passage	9.2	0	0	1	0	0	1
passage	9.5	0	0	1	0	0	1
passage	9.7	0	0	3	6	5	14
passage	9.8	0	1	1	0	0	2
passage	9.9	0	1	0	0	0	1
passer	1	0	0	0	0	1	1
passer	1.1.1	155	0	13	2	4	174
passer	1.1.1.1	0	0	1	0	0	1
passer	1.1.1.2	3	0	0	0	4	7
passer	1.1.2	272	10	33	18	21	354
passer	1.1.2.1	0	0	1	0	0	1
passer	1.1.2.2	0	0	2	0	0	2
passer	1.1.2.3	0	0	1	0	7	8
passer	1.1.2.4	1	0	1	1	1	4
passer	1.1.3	49	1	12	3	5	70
passer	1.1.4	4	0	9	2	1	16
passer	1.1.5	12	0	1	0	1	14

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
passer	1.1.6	1	0	4	0	6	11
passer	1.10.1	32	0	3	1	1	37
passer	1.10.10	0	0	2	0	0	2
passer	1.10.11	3	0	3	1	0	7
passer	1.10.12	2	0	0	1	1	4
passer	1.10.13	2	1	0	2	2	7
passer	1.10.15	0	0	2	1	0	3
passer	1.10.16	0	0	1	5	3	9
passer	1.10.17	0	0	1	0	0	1
passer	1.10.18	1	0	1	0	1	3
passer	1.10.19	1	0	1	0	0	2
passer	1.10.2	21	1	9	2	7	40
passer	1.10.2.1	0	0	2	0	1	3
passer	1.10.3	2	27	22	14	8	73
passer	1.10.4	1	2	3	5	0	11
passer	1.10.5	1	0	1	4	2	8
passer	1.10.5.1	0	0	1	4	3	8
passer	1.10.6	4	4	6	2	6	22
passer	1.10.7.1	16	0	2	1	1	20
passer	1.10.7.2	1	2	1	0	0	4
passer	1.10.7.3	1	0	1	0	1	3
passer	1.10.7.4	0	0	2	0	0	2
passer	1.10.7.5	1	0	1	0	0	2
passer	1.10.8.1	0	0	5	1	0	6
passer	1.10.8.2	5	0	2	0	0	7
passer	1.10.8.3	3	0	0	0	1	4
passer	1.10.8.4	1	0	0	0	0	1
passer	1.10.9	1	0	2	0	0	3
passer	1.11	0	3	0	0	3	6
passer	1.11.1	4	0	2	4	8	18
passer	1.12	5	0	1	1	0	7
passer	1.13	0	0	1	0	0	1
passer	1.13.1	2	0	3	17	1	23
passer	1.14	0	0	5	1	1	7
passer	1.14.1	0	0	1	0	0	1
passer	1.15.1	0	0	1	1	1	3
passer	1.15.2	0	1	0	0	0	1
passer	1.15.3	0	1	2	2	0	5
passer	1.15.4	0	0	1	0	0	1
passer	1.15.5	8	3	6	1	8	26
passer	1.15.6	0	1	0	0	0	1
passer	1.15.7	1	0	0	2	0	3
passer	1.15.8	0	0	2	1	0	3
passer	1.15.9	1	0	0	0	0	1
passer	1.2.1	12	0	18	4	4	38
passer	1.2.2	127	8	58	20	9	222
passer	1.3.1	4	11	44	51	49	159
passer	1.3.2	2	4	29	18	21	74
passer	1.3.3	2	0	1	5	1	9
passer	1.3.4	0	0	1	0	0	1
passer	1.4.1	24	1	17	14	11	67
passer	1.4.2	7	2	6	2	5	22

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
passer	1.5	0	0	1	1	1	3
passer	1.6	0	0	0	0	1	1
passer	1.7.1	3	0	6	1	4	14
passer	1.7.2	19	0	7	2	2	30
passer	1.7.2.1	0	0	1	1	1	3
passer	1.7.3	1	0	0	0	0	1
passer	1.7.4	1	0	0	0	0	1
passer	1.7.5	3	0	0	0	0	3
passer	1.7.6	3	0	2	0	0	5
passer	1.7.7	1	0	0	0	0	1
passer	1.8	28	43	130	115	89	405
passer	1.8.1	1	0	1	0	0	2
passer	1.9	4	0	6	6	3	19
passer	2.1	78	11	100	27	61	277
passer	2.1.1	0	0	10	11	20	41
passer	2.2	18	0	2	2	1	23
passer	2.3	0	2	2	0	0	4
passer	2.4	6	0	1	0	0	7
passer	2.5	23	3	12	15	6	59
passer	2.6	3	0	0	0	0	3
pied	1.1.1	307	1	41	7	5	361
pied	1.1.1.1	5	0	3	0	0	8
pied	1.1.1.2	4	0	1	0	0	5
pied	1.1.2	9	0	0	0	0	9
pied	1.1.3	3	0	0	0	0	3
pied	1.1.3.1	1	0	0	0	0	1
pied	1.1.3.2	2	1	0	1	0	4
pied	1.2.1	46	0	4	0	0	50
pied	1.2.2	49	3	13	3	2	70
pied	1.2.3	0	0	1	0	0	1
pied	1.3.1	33	6	11	1	1	52
pied	1.3.2	2	0	0	0	0	2
pied	1.4.1	4	35	21	1	9	70
pied	1.4.12	0	0	0	0	1	1
pied	1.4.15	12	0	0	1	0	13
pied	1.4.16	1	0	1	0	0	2
pied	1.4.19	5	2	1	1	0	9
pied	1.4.2	0	4	1	0	1	6
pied	1.4.20	0	0	1	1	0	2
pied	1.4.23	0	0	1	0	0	1
pied	1.4.24	0	0	2	0	0	2
pied	1.4.25	0	0	4	2	0	6
pied	1.4.26	1	0	4	0	0	5
pied	1.4.28	0	0	6	0	0	6
pied	1.4.29	2	0	0	0	0	2
pied	1.4.3	0	1	0	1	1	3
pied	1.4.32	1	0	0	0	0	1
pied	1.4.33	0	0	1	0	0	1
pied	1.4.34	1	0	0	0	0	1
pied	1.4.35	0	0	1	0	0	1
pied	1.4.36	1	0	2	0	0	3
pied	1.4.37	0	0	2	0	0	2

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
pied	1.4.38	6	0	1	0	0	7
pied	1.4.39	6	0	0	0	0	6
pied	1.4.4	0	6	0	7	3	16
pied	1.4.40	0	0	1	0	0	1
pied	1.4.41	0	0	1	0	0	1
pied	1.4.42	1	0	0	0	0	1
pied	1.4.43	2	0	0	0	0	2
pied	1.4.44	0	0	1	0	0	1
pied	1.4.45	1	0	0	0	0	1
pied	1.4.46	4	0	2	1	0	7
pied	1.4.47	0	0	2	1	0	3
pied	1.4.48	0	0	1	0	1	2
pied	1.4.49	0	0	4	0	0	4
pied	1.4.5	0	0	3	0	1	4
pied	1.4.50	1	0	0	0	0	1
pied	1.4.51	1	0	0	0	0	1
pied	1.4.52	1	0	0	0	0	1
pied	1.4.53	0	0	1	0	0	1
pied	1.4.54	0	0	8	2	2	12
pied	1.4.55	3	0	0	1	1	5
pied	1.4.56	0	0	0	0	1	1
pied	1.4.57	2	0	0	0	0	2
pied	1.4.58	0	0	0	1	0	1
pied	1.4.59	1	0	0	0	0	1
pied	1.4.6	0	0	2	0	0	2
pied	1.4.60	1	0	0	0	0	1
pied	1.4.61	0	0	1	0	0	1
pied	2.1	147	0	1	0	0	148
pied	2.1.1	22	0	0	0	0	22
pied	2.2	1	0	0	0	0	1
plein	1.1.1	30	0	3	2	1	36
plein	1.1.1.1	1	0	0	0	0	1
plein	1.1.2	7	2	0	0	3	12
plein	1.1.3	2	0	0	3	1	6
plein	1.2.2	1	0	0	0	0	1
plein	1.2.3	0	0	1	4	1	6
plein	1.2.4	0	0	2	0	0	2
plein	1.4.1	52	0	5	0	0	57
plein	1.4.2	13	0	5	0	2	20
plein	2.1	18	0	2	0	0	20
plein	2.2	17	0	0	3	2	22
plein	2.2.1	0	0	1	0	0	1
plein	3.1	13	46	6	18	15	98
plein	3.1.1	9	0	1	1	0	11
plein	3.1.3	0	3	0	0	0	3
plein	3.1.4	0	0	0	2	1	3
plein	3.2	0	0	9	0	0	9
plein	4.1.1	65	0	2	0	1	68
plein	4.1.2	92	0	9	12	7	120
plein	4.1.3	2	0	0	0	0	2
plein	4.2.1	1	0	0	1	0	2
plein	4.2.2	2	0	0	2	0	4

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
plein	4.3.1	1	0	0	0	0	1
plein	5	18	1	6	4	0	29
plein	5.1	1	0	0	0	0	1
plein	5.2	0	13	9	2	3	27
plein	5.3	0	1	3	2	2	8
plein	6.1.1	27	2	12	2	2	45
plein	6.1.2	10	0	1	0	0	11
plein	6.1.2.1	0	0	1	0	0	1
plein	6.1.3	47	1	50	30	16	144
plein	6.1.4	0	0	1	0	0	1
plein	6.2	42	4	8	1	1	56
plein	6.3	2	0	0	0	1	3
plein	7.1	0	3	6	1	3	13
populaire	1.1	11	13	52	35	108	219
populaire	1.2	1	0	18	16	17	52
populaire	1.3	1	11	10	11	17	50
populaire	2	5	3	18	15	9	50
populaire	3	0	15	43	10	18	86
porter	1.1	119	0	18	8	9	154
porter	1.1.1	2	0	0	0	0	2
porter	1.10	68	1	3	7	2	81
porter	1.10.1	1	8	4	4	0	17
porter	1.10.2	2	0	9	0	0	11
porter	1.10.3	0	0	1	0	0	1
porter	1.10.4	4	0	0	0	0	4
porter	1.10.4.1	0	0	2	0	0	2
porter	1.10.5	0	0	2	0	1	3
porter	1.10.6	4	0	11	1	2	18
porter	1.10.7	1	0	0	1	0	2
porter	1.10.8	0	0	2	0	0	2
porter	1.10.9	4	0	4	6	2	16
porter	1.11.1	23	0	0	0	0	23
porter	1.11.2	3	0	0	0	0	3
porter	1.11.3	1	0	0	0	0	1
porter	1.12	15	30	37	14	18	114
porter	1.13	5	41	26	44	39	155
porter	1.13.1	0	0	4	4	1	9
porter	1.13.2	0	0	1	6	1	8
porter	1.14	0	27	2	7	5	41
porter	1.15	7	83	17	43	3	153
porter	1.16	2	109	4	12	1	128
porter	1.17	0	0	0	1	0	1
porter	1.17.1	0	2	5	5	1	13
porter	1.18	0	2	1	2	3	8
porter	1.18.1	21	229	79	198	163	690
porter	1.18.2	3	1	1	0	0	5
porter	1.19	0	0	0	4	0	4
porter	1.2.1	19	0	1	0	1	21
porter	1.2.2	8	3	11	5	1	28
porter	1.2.2.1	0	0	0	0	1	1
porter	1.2.3	1	0	0	0	0	1
porter	1.20	1	0	1	0	0	2

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
porter	1.21	4	0	2	1	0	7
porter	1.22	0	0	1	0	0	1
porter	1.23	0	0	0	0	1	1
porter	1.3	16	1	18	35	63	133
porter	1.3.1	0	0	1	3	3	7
porter	1.4	15	10	10	8	4	47
porter	1.5	8	6	23	5	12	54
porter	1.6	136	2	32	3	37	210
porter	1.6.1	1	0	0	0	0	1
porter	1.6.2	0	0	1	0	0	1
porter	1.7	0	0	0	1	0	1
porter	1.7.1	2	0	1	0	0	3
porter	1.8	3	0	2	2	1	8
porter	1.9	10	0	1	1	0	12
porter	2.1	7	1	1	4	16	29
porter	2.2	0	0	5	0	0	5
porter	3.1	9	0	4	4	0	17
porter	3.2	3	0	0	0	0	3
porter	4.1	13	0	7	1	2	23
porter	4.2	4	3	9	1	2	19
porter	4.3.1	19	0	0	0	1	20
porter	4.3.2	0	0	2	3	3	8
porter	4.3.3	0	0	1	0	0	1
porter	4.4	1	0	1	0	11	13
porter	4.5	1	0	0	0	0	1
poursuivre	1.1	3	38	5	36	17	99
poursuivre	1.2	0	0	1	0	0	1
poursuivre	1.2.1	13	171	68	66	36	354
poursuivre	1.2.2	61	2	32	11	9	115
poursuivre	1.2.3	0	2	1	0	1	4
poursuivre	1.2.3.1	1	0	2	0	1	4
poursuivre	1.2.4	0	0	0	1	1	2
poursuivre	1.3.1	9	0	1	2	0	12
poursuivre	1.3.2	16	0	1	0	0	17
poursuivre	1.3.2.1	4	0	0	0	1	5
poursuivre	1.3.3	3	0	2	1	0	6
poursuivre	1.3.4	2	0	0	2	0	4
poursuivre	1.4.1	2	12	21	118	2	155
poursuivre	1.4.2	0	3	2	57	1	63
poursuivre	2.1	5	43	33	30	24	135
poursuivre	2.2	2	0	0	0	0	2
présenter	1.1	27	173	19	101	204	524
présenter	1.2	23	0	11	11	3	48
présenter	1.2.1	6	8	40	43	58	155
présenter	1.2.2	21	536	93	99	110	859
présenter	1.2.2.1	3	0	4	0	0	7
présenter	1.2.3	22	16	8	3	1	50
présenter	1.2.4	4	0	0	1	0	5
présenter	1.2.5	0	0	15	0	2	17
présenter	1.2.5.1	0	0	1	0	1	2
présenter	1.3	7	5	47	102	102	263
présenter	1.4	5	11	16	1	4	37

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
présenter	1.5	0	2	2	0	0	4
présenter	1.6.1	3	0	0	0	0	3
présenter	1.6.2	1	0	0	0	0	1
présenter	2.1	2	1	5	0	1	9
présenter	2.1.1	0	0	4	0	0	4
présenter	2.2	31	1	14	3	4	53
présenter	2.3	26	21	10	21	23	101
rendre	1.1	68	0	26	5	6	105
rendre	1.1.1	0	0	0	1	0	1
rendre	1.1.2	1	0	0	0	0	1
rendre	1.1.3	0	0	1	0	0	1
rendre	1.10	0	0	0	2	1	3
rendre	1.11	0	0	0	0	1	1
rendre	1.2	24	0	4	6	1	35
rendre	1.2.1	25	2	9	21	7	64
rendre	1.2.2	8	3	7	9	0	27
rendre	1.3	8	53	26	30	7	124
rendre	1.3.1	19	12	16	76	108	231
rendre	1.3.2	5	0	0	1	4	10
rendre	1.4	14	4	17	13	5	53
rendre	1.5	4	2	11	6	1	24
rendre	1.6	114	160	135	231	283	923
rendre	1.7.1	1	0	0	0	0	1
rendre	1.7.2	4	0	0	0	0	4
rendre	1.7.3	4	0	0	0	1	5
rendre	1.7.4	0	0	2	0	1	3
rendre	1.8	3	0	1	7	6	17
rendre	1.9	1	0	0	0	0	1
rendre	2.1	49	37	69	20	10	185
rendre	2.2	12	0	5	1	2	20
rendre	2.3	2	0	3	1	0	6
rendre	2.3.1	1	0	3	3	1	8
rendre	2.4	9	11	4	6	12	42
rendre	2.5	19	9	27	19	21	95
restauration	1.1	1	9	10	10	5	35
restauration	1.2	0	0	2	0	0	2
restauration	1.3	1	0	2	8	0	11
restauration	2	0	33	8	1	3	45
restauration	3	0	7	3	0	1	11
régulier	1.1	1	3	7	4	3	18
régulier	1.1.1	3	0	1	0	0	4
régulier	1.2	0	0	0	0	1	1
régulier	1.3	1	0	0	0	0	1
régulier	1.4	0	0	0	0	1	1
régulier	1.5	1	0	0	0	0	1
régulier	2.1	14	2	4	6	6	32
régulier	2.2	0	38	4	9	8	59
régulier	2.3	1	20	7	7	7	42
régulier	2.4	10	0	1	1	5	17
régulier	2.5	4	0	0	1	0	5
répondre	1.1	1386	190	158	120	127	1981
répondre	1.1.1	1	0	0	0	0	1

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
répondre	1.2	8	10	0	1	1	20
répondre	1.3	0	4	0	1	2	7
répondre	1.4	5	191	57	102	90	445
répondre	1.5.1	5	2	12	19	10	48
répondre	1.5.2	2	0	3	3	1	9
répondre	1.6	1	0	0	0	4	5
répondre	2	5	0	2	2	4	13
sain	1.1.1	1	1	1	6	26	35
sain	1.1.1.1	3	0	0	0	1	4
sain	1.1.2	1	0	0	2	4	7
sain	1.1.3	0	2	0	0	0	2
sain	1.2	0	3	0	0	0	3
sain	1.3	1	0	1	0	10	12
sain	1.4	2	0	0	0	0	2
sain	2	7	11	16	9	9	52
sain	3.1	0	7	1	1	0	9
sain	3.2	1	1	1	0	0	3
secondaire	1	0	1	1	0	0	2
secondaire	2	5	13	15	7	8	48
secondaire	3.1	2	2	8	14	3	29
secondaire	3.2	2	25	14	36	28	105
secondaire	3.3	0	0	0	8	3	11
sensible	1.1	5	0	2	2	1	10
sensible	1.2	26	0	2	7	18	53
sensible	1.3	1	0	0	1	0	2
sensible	1.4	0	54	11	16	5	86
sensible	1.5	11	26	21	47	22	127
sensible	2.1	6	11	24	35	22	98
sensible	2.2	0	2	5	1	7	15
sensible	2.3.1	1	0	0	2	8	11
sensible	2.3.2	1	2	0	0	8	11
sensible	2.3.3	0	0	1	0	4	5
sensible	2.4	0	0	0	0	7	7
simple	1.1	104	7	75	118	100	404
simple	1.1.1	1	1	0	0	0	2
simple	1.2	54	26	64	144	146	434
simple	1.2.1	2	9	4	10	5	30
simple	1.3.1	12	4	0	2	14	32
simple	1.3.2	5	2	0	0	9	16
simple	1.3.2.1	12	0	0	0	1	13
simple	1.4	11	1	4	1	0	17
simple	1.5	2	0	1	24	0	27
simple	2.1	5	0	2	3	0	10
simple	2.2	9	4	13	12	18	56
simple	2.3	6	0	1	0	0	7
simple	2.4	0	0	0	0	1	1
simple	2.5	0	0	1	0	1	2
solution	1	14	185	139	351	132	821
solution	2	0	3	5	0	30	38
solution	3.1	0	11	1	0	0	12
solution	3.2	0	0	0	0	9	9
station	1.1	0	7	49	0	7	63

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
station	1.2	1	21	8	2	53	85
station	2	0	2	9	7	2	20
station	3	4	2	7	1	0	14
station	4	0	38	4	1	0	43
station	4.1	0	9	2	1	1	13
station	4.2	0	0	0	22	0	22
station	6	2	0	0	1	3	6
strict	1.1	0	50	11	23	16	100
strict	1.2	2	0	0	1	0	3
strict	2	0	20	7	12	5	44
strict	3	3	2	4	19	14	42
strict	3.1	0	2	2	4	3	11
strict	3.2	0	3	1	0	0	4
strict	3.3	0	3	1	0	0	4
strict	4	0	0	2	1	4	7
strict	5	0	0	4	1	0	5
suspension	1	0	48	13	2	5	68
suspension	2	0	0	2	1	0	3
suspension	3	0	0	1	14	8	23
suspension	4	0	10	1	0	4	15
suspension	5	0	0	0	0	1	1
sûr	1.1.1	27	0	17	6	6	56
sûr	1.1.2	9	1	0	2	0	12
sûr	1.2.1	11	0	1	1	2	15
sûr	1.2.1.1	0	0	1	0	0	1
sûr	1.2.2	5	0	3	1	2	11
sûr	1.3	40	2	31	6	5	84
sûr	1.4	0	3	7	3	9	22
sûr	1.5.1	14	8	126	57	91	296
sûr	1.5.2	13	3	8	7	9	40
sûr	1.5.3	4	0	0	0	0	4
sûr	1.5.4	1	0	0	0	0	1
sûr	2	17	31	12	11	7	78
sûr	3.1	5	0	2	1	1	9
sûr	3.2	2	0	5	2	7	16
tirer	1.1.1	64	0	21	5	1	91
tirer	1.1.2	2	0	0	0	0	2
tirer	1.1.3	13	0	9	0	0	22
tirer	1.10	0	1	0	0	0	1
tirer	1.11	0	0	1	0	0	1
tirer	1.12	0	0	3	0	0	3
tirer	1.2.1	51	0	3	10	3	67
tirer	1.2.2	3	0	2	0	5	10
tirer	1.2.3	9	0	0	0	0	9
tirer	1.2.4	0	0	3	1	0	4
tirer	1.4.1	66	61	46	69	48	290
tirer	1.4.2	6	2	5	10	17	40
tirer	1.4.3	19	12	17	28	26	102
tirer	1.4.3.1	13	6	11	11	4	45
tirer	1.4.4	5	0	0	0	1	6
tirer	1.4.4.1	1	0	0	0	0	1
tirer	1.4.5	4	2	3	9	16	34

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
tirer	1.4.6.1	85	0	2	0	1	88
tirer	1.4.6.2	11	0	0	2	3	16
tirer	1.4.7	5	2	2	0	4	13
tirer	1.5	2	0	8	7	3	20
tirer	1.6.1	0	0	1	0	0	1
tirer	1.6.2	1	0	0	0	0	1
tirer	1.7	1	0	1	0	0	2
tirer	1.8	17	0	0	0	1	18
tirer	1.9	4	0	0	0	0	4
tirer	2.1	25	1	6	2	0	34
tirer	2.1.1	11	0	1	2	0	14
tirer	2.2.1	0	0	1	0	0	1
tirer	2.2.2	1	0	0	0	0	1
tirer	2.3	10	0	0	0	3	13
tirer	3.11	0	0	5	1	1	7
tirer	3.12	1	1	2	0	0	4
tirer	3.13	0	0	1	0	0	1
tirer	3.15	0	0	2	1	1	4
tirer	3.16	0	0	1	0	0	1
tirer	3.17	6	0	0	0	0	6
tirer	3.18	0	0	0	0	3	3
tirer	3.19	1	0	0	0	0	1
tirer	3.20	0	0	1	0	0	1
tirer	3.21	2	0	0	0	0	2
tirer	3.22	0	0	0	0	1	1
tirer	3.3	1	0	0	0	0	1
tirer	3.4	0	3	4	0	1	8
tirer	3.5	0	2	0	0	0	2
tirer	3.6	1	0	0	0	0	1
tirer	3.7	4	0	0	1	0	5
traditionnel	1	0	78	74	127	121	400
traditionnel	2	0	3	31	7	6	47
utile	1.1	27	37	16	45	29	154
utile	1.2	11	16	10	29	11	77
utile	1.3	1	17	5	7	10	40
utile	1.4	2	11	2	9	14	38
utile	1.5	0	1	2	1	1	5
utile	1.6.1	0	7	0	0	2	9
utile	1.6.2	0	1	0	3	0	4
utile	1.6.3	0	12	1	1	0	14
utile	2	11	0	3	3	1	18
vaste	1.1	60	13	7	0	12	92
vaste	1.2	36	1	13	1	1	52
vaste	1.3	5	2	8	4	5	24
vaste	2	0	42	0	0	0	42
vaste	2.1	20	0	27	63	46	156
vaste	2.2	0	0	0	2	0	2
venir	1.1.1	11	45	48	37	18	159
venir	1.1.2	152	35	103	101	91	482
venir	1.1.2.1	1	0	0	0	0	1
venir	1.1.3	48	0	8	5	5	66
venir	1.1.4	6	0	10	5	11	32

Suite sur la page suivante...

Suite de la page précédente...

Vocables	**Lexies**	ABU	JOC	MON	OUV	PER	**TOTAL**
venir	1.1.5	1	0	0	0	1	2
venir	1.2.1	134	15	74	12	16	251
venir	1.2.2	60	3	57	56	61	237
venir	1.2.3	14	4	3	6	10	37
venir	1.3	318	99	233	72	225	947
venir	1.4.1	584	2	89	28	17	720
venir	1.4.1.1	15	0	0	0	0	15
venir	1.4.2	357	4	144	39	22	566
venir	1.4.2.1	2	0	0	0	0	2
venir	1.4.3	22	1	8	5	0	36
venir	1.4.4	2	0	0	0	0	2
venir	1.5.1	19	6	12	12	6	55
venir	1.5.1.1	9	4	12	14	36	75
venir	1.5.2	0	10	5	0	0	15
venir	2.1	8	3	6	2	4	23
venir	2.10	1	0	0	0	0	1
venir	2.11	0	0	0	1	0	1
venir	2.12	4	0	0	0	0	4
venir	2.13	1	0	0	0	0	1
venir	2.14	0	0	0	2	0	2
venir	2.15	0	0	1	0	0	1
venir	2.2	2	0	1	0	0	3
venir	2.3	0	0	2	0	0	2
venir	2.5	20	10	5	3	6	44
venir	2.6	2	1	2	1	0	6
venir	2.7	1	0	0	1	0	2
venir	2.8	4	0	0	0	0	4
venir	2.9	0	0	0	3	0	3
vol	1.1	17	1	0	1	32	51
vol	1.2	0	1	0	0	0	1
vol	2	14	19	14	13	5	65
vol	3.1	3	63	17	5	24	112
vol	3.2	16	10	1	1	3	31
vol	3.3	2	0	0	0	1	3
vol	4.1	1	0	0	0	0	1
vol	4.3	9	0	2	1	0	12
vol	4.4	1	0	0	0	0	1
vol	4.5	0	0	1	0	0	1
économie	1.1	1	106	162	62	126	457
économie	1.2	2	51	58	34	19	164
économie	1.2.1	1	0	0	48	4	53
économie	1.3.1	0	20	8	5	9	42
économie	1.3.2	0	0	4	0	3	7
économie	2.1	26	33	35	51	18	163
économie	2.1.1	0	0	7	10	9	26
économie	2.2	9	0	1	1	0	11
économie	2.3	0	3	0	0	0	3
économie	3	0	0	0	4	0	4
TOTAL		13 428	10 292	9141	10 090	10 845	53 796

Tableau D.1 – Fréquence de chacune des occurrences des lexies des 60 vocables dans chacun des sous-corpus ainsi que dans tout le corpus (colonne *TOTAL*).

Annexe E

Informations relatives aux étiquettes et à leur codage

E.1 Fréquences des étiquettes morphosyntaxiques

Le tableau E.1 recense les trois types d'étiquettes morphosyntaxiques observées sur l'ensemble des mots du corpus en précisant leur fréquence. Les étiquettes *ems* constituent des sous-ensembles des étiquettes simplifiées *smallems*. Les étiquettes *tgb* sont encore plus précises que les étiquettes *ems* mais ne constituent pas exactement des sous-ensembles des étiquettes *ems*.

Les étiquettes effectivement générées par le logiciel Cordial Analyseur ne correspondent pas tout à fait à ce qui est indiqué dans le manuel d'utilisation. Il est donc préférable de se reporter aux tableaux de cette section et des trois sections suivantes qui détaillent la signification de ces étiquettes. Nous avons remarqué que les étiquettes *ems* et *tgb* ne sont pas toujours cohérentes entre elles.

Étiquette smallems	**Étiquette ems**	**Étiquette tgb**	**Fréquence**	**Exemples d'occurrences dans le corpus**
ADJ	ADJFP	Afcfp	88	meilleures
		Afpfp	43 389	additives affaiblies antinationales
	ADJFS	Afcfs	273	meilleure
		Afpfs	79 093	admise affreuse agonisante alignée
	ADJHFS	Afpfs	538	haute hachée haeckelienne hagarde
	ADJHMS	Afpms	378	hagard haut hérissé allemand fini
	ADJHSIG	Afp.s	35	haïssable hittite hiérarchique
		Afpfs	54	haïssable hiérarchique
		Afpms	14	hiérarchique
	ADJIND	Dt-..-	1823	la de des du force plupart
		Dt-.p-	10 321	plusieurs quelques autres quelconques
		Dt-.s-	3026	chaque quelque chaque quelconque
		Dt-fp-	2739	certaines différentes diverses maintes
		Dt-fs-	4052	aucune mainte nulle telle toute
		Dt-mp-	3230	certains différents divers tels aucuns
		Dt-ms-	3692	aucun nul tel tout aucun certain
	ADJINT	Ai-fp-	1242	quelles
		Ai-fs-	1247	quelle

Suite sur la page suivante...

Suite de la page précédente...

smallems	ems	tgb	fréq	Exemples
		Ai-mp-	700	quels
		Ai-ms-	1051	quel
	ADJINV	Afp..	2583	améthyste arc avant bio boeuf
		Afpfp	159	nord standard sud antipollution
		Afpfs	399	café est nature nord ouest pêche
		Afpmp	135	nord air antipollution antivol argent
		Afpms	512	est nature nord ouest pie pêche rosat
	ADJMIN	Afpm.	5573	affreux acquis albanais anglais animaux
		Afpmp	2437	albanais contentieux français mis
		Afpms	3269	astucieux anglais boiteux chaleureux
	ADJMP	Afcmp	153	meilleurs
		Afpmp	40 134	additifs adversatifs affaiblis ajoutés
	ADJMS	Afcms	163	meilleur
		Afpms	76 695	abasourdi absent accusé accéléré actuel
	ADJNUM	Mc..	65 845	2 1 2 20 21 3 32 37 4 5 6 62
		Mc.p	22 530	cent cinq cinquante deux deux dix
		Mcfs	3198	0 10 15 16 30 40 5 50 51 57
		Mcms	216	un
	ADJORD	Ao-..	8953	100e 100ème 101e 102e 103e
		Ao-m.	50	cinquième premier premier cinquième
	ADJPIG	Afc.p	45	pires moindres
		Afcfp	17	moindres pires
		Afcmp	36	moindres pires
		Afp.p	17 996	amateurs analytiques antarctiques
		Afpfp	13 336	atlantiques autres britanniques difficiles
		Afpmp	10 533	analytiques autres critiques
	ADJSIG	Afc.s	106	pire moindre
		Afcfs	142	moindre pire
		Afcms	108	moindre pire
		Afp.s	37 016	acerbe admirable agricole anarchiste
		Afpfs	27 603	admirable astronomique autre brave
		Afpms	20 761	admirable aimable automobile aveugle
ADV	ADV	Rgc	5027	mieux moins pis
		Rgn	60 504	combien comme guère jamais non
		Rgp	172 229	absolument accessoirement
		Rpn	66 702	ne
COO	COO	Cc	152 528	car donc et mais ni néanmoins or ou
DET	DETDEM	Dd-.p-	13 270	ces
		Dd-fs-	16 265	cette
		Dd-ms-	17 441	ce cet
	DETDFS	Da-f-d	1	l
		Da-fs-d	175 558	la bas
		Da-ms-d	119 060	l
	DETDMS	Da-m-d	60	the
		Da-ms-d	293 349	au du le quant
	DETDPIG	Da-.p-d	114 518	aux le les quant
		Da-.p-i	98 275	des
	DETIFS	Da-fs-i	53 640	une
	DETIMS	Da-ms-i	54 121	un
	DETPOSS	As-.p	1	nôtres

Suite sur la page suivante...

Suite de la page précédente...

smallems	ems	tgb	fréq	Exemples
		As-.s	6	nôtre vôtre
		As-fp	6	siennes
		As-fs	24	mienne sienne tienne
		As-mp	6	siens
		As-ms	10	mien mien sien tien
		Ds1.p.	1327	nos
		Ds1.ps	712	mes
		Ds1.s.	1973	notre
		Ds1.ss	2097	mon
		Ds1fss	1208	ma
		Ds2.ps	502	tes vos
		Ds2.ss	1046	ton votre
		Ds2fss	288	ta
		Ds3.pp	4948	leurs
		Ds3.ps	9940	ses
		Ds3.sp	8332	leur
		Ds3.ss	13 959	son
		Ds3fss	10 384	sa
INT	INT	I	2143	adieu ah alerte allo amen attention aïe
NCOM	NCFIN	Ncf.	5655	annales bellis dame fois fulica mimesis
	NCFP	Ncfp	143 983	abeilles absences académies acceptations
	NCFS	Ncfs	432 358	abeille abolition abrogation absence
	NCHFS	Ncfs	1012	hache haine halloween halte harpe
	NCHMIN	Ncm.	188	hongrois héros harnais hautbois hic
	NCHMS	Ncms	1221	commissariat hallier handicap
	NCHSIG	Nc.s	30	holding huitième harpiste holding
	NCI	Nc..	41 856	+ , / 0 25 1 50 8 1 10 11
	NCMIN	Ncm.	188 268	0 25 1 17 2 20 27 3 36 4 40
	NCMP	Ncmp	161 850	abandons abattoirs abbés aboiements
	NCMS	Ncms	364 162	abaissement abandon abattage abbé
	NCPIG	Nc.p	13 912	aides architectes affichistes afrikaners
	NCSIG	Nc.s	15 584	antique actionnaire adversaire aide alto
NHMIN	NHMIN	Ncmp	356	hacheurs hadji hambourgeois haras
		Npmp	15	alpes hautes
NPRO	NPFIN	Npf.	81	alfort ariane flandres jaguar laffitte
	NPFP	Npfp	542	alpes atlantiques abruzzes alpes andes
	NPFS	Npfs	28 930	ademe afrique agirc aima algérie
	NPHFS	Npfs	225	garonne haute hague hanse haute
	NPHMS	Npms	410	halley hamburger harlem hart harvard
	NPHSIG	Np.s	17 029	ariège aude abondance acre adelphe
	NPI	Np..	70 045	( ) * , / 1 3 1 10 1043a 11 111
	NPMIN	Npm.	469	adams adorno airbus albigeois anges
	NPMP	Npmp	3163	abymes america anglo argonautes bas
	NPMS	Npms	72 589	adam alain alberti alessandri almond
	NPPIG	Np.p	236	airlines azéris bouches bahamas
	NPSIG	Np.s	17 680	alpes apt ardenne aveyron abbaye
PCTFAIB	PCTFAIB	Ypc	42 401	) -] },
		Ypo	42 068	([{
		Ypw	356 608	! , - . : ; ?
PCTFORTE	PCTFORTE	Yps	252 104	! ’ - . : ; ?

Suite sur la page suivante...

Suite de la page précédente...

smallems	ems	tgb	fréq	Exemples
PRODE	PDP	Pd-fp-	383	celles ci là
		Pd-fpa	327	celles ci là
		Pd-fpd	438	celles ci là
		Pd-fpn	754	celles ci là
		Pd-mp-	610	ceux ci là
		Pd-mpa	615	ceux ci là
		Pd-mpd	925	ceux ci là
		Pd-mpn	1420	ceux ci là
	PDS	Pd-..-	3060	ce celui ci
		Pd-..a	1780	ce
		Pd-..d	3098	ce
		Pd-..n	25 703	ce celui ci
		Pd-.s-	633	ceci cela ça
		Pd-.sa	499	ceci cela ça
		Pd-.sd	648	ceci cela ça
		Pd-.sn	2793	ceci cela ça
		Pd-f.n	2	ce
		Pd-fs-	1123	celle ci là
		Pd-fsa	769	celle ci là
		Pd-fsd	968	celle ci là
		Pd-fsn	1721	celle ci là
		Pd-ms-	1053	celui ci là
		Pd-msa	821	celui ci là
		Pd-msd	1064	celui ci là
		Pd-msn	1876	celui ci là
PROIN	PIFP	Pi-fp-	160	certaines telles toutes certaines
		Pi-fpa	80	certaines quelques toutes unes
		Pi-fpd	33	certaines quelques toutes unes
		Pi-fpn	167	certaines quelques telles toutes unes
	PIFS	Pi-..-	2	certaine
		Pi-..a	1	certaine
		Pi-..d	2	certaine
		Pi-..n	2	certaine
		Pi-fpd	3	une
		Pi-fs-	1023	aucune toute une chacune nulle telle
		Pi-fsa	538	une aucune chacune quelqu telle toute
		Pi-fsd	678	une aucune chacune nulle quelqu telle
		Pi-fsn	937	aucune chacune telle toute une nulle
	PII	Pi-.p-	26	la plupart
		Pi-.pa	8	la plupart
		Pi-.pd	122	la plupart
		Pi-.pn	92	la plupart
		Pi-.s-	529	rien
		Pi-.sa	923	rien
		Pi-.sd	273	rien
		Pi-.sn	504	rien
	PIMP	Pi-..-	6	autres
		Pi-..a	6	autres
		Pi-..n	6	autres
		Pi-.p-	459	autres

Suite sur la page suivante...

Suite de la page précédente...

smallems	**ems**	**tgb**	**fréq**	**Exemples**
		Pi-.pa	195	autres
		Pi-.pd	576	autres
		Pi-.pn	774	autres
		Pi-mp-	537	certains quelques tous tels uns
		Pi-mpa	279	tous certains quelques tels uns
		Pi-mpd	449	certains quelques tels tous uns
		Pi-mpn	846	certains quelques tels tous uns
	PIMS	Pi-.s-	129	quelqu un
		Pi-.sa	177	quelqu un
		Pi-.sd	153	quelqu un
		Pi-.sn	267	quelqu un
		Pi-fpd	6	un
		Pi-ms-	2398	aucun autrui chacun nul tel tout un
		Pi-msa	1315	aucun chacun tel tout un autrui
		Pi-msd	1555	chacun tel tout un aucun autrui
		Pi-msn	2788	aucun autrui chacun nul tel tout un
	PIPIG	Pi-..-	2	albert
		Pi-.p-	546	autres plusieurs mêmes plusieurs
		Pi-.pa	201	autres mêmes plusieurs
		Pi-.pd	515	autres mêmes plusieurs
		Pi-.pn	273	plusieurs autres mêmes
	PISIG	Pi-.s-	650	autre même
		Pi-.sa	256	autre même
		Pi-.sd	848	autre même quiconque
		Pi-.sn	331	autre même
		Pi-ms-	58	personne
		Pi-msa	74	personne
		Pi-msd	90	kilomètre personne
		Pi-msn	180	personne
PROPO	PP	Ps1.p-.	6	nôtres
		Ps1.pa.	5	nôtres
		Ps1.pd.	6	nôtres
		Ps1.pn.	5	nôtres
		Ps1.s-.	24	nôtre
		Ps1.sa.	7	nôtre
		Ps1.sd.	13	nôtre
		Ps1.sn.	11	nôtre
		Ps1fp-s	1	tiennes
		Ps1fpds	1	tiennes
		Ps1fs-s	8	mienne
		Ps1fsas	7	mienne
		Ps1fsds	6	mienne
		Ps1fsns	6	mienne
		Ps1mp-s	5	miennes miens
		Ps1mpas	6	miennes miens
		Ps1mpds	3	miens
		Ps1mpns	2	miens
		Ps1ms-s	9	mien
		Ps1msas	7	mien
		Ps1msds	3	mien

Suite sur la page suivante...

Suite de la page précédente...

smallems	ems	tgb	fréq	Exemples
		Ps1msns	6	mien
		Ps2.p-.	5	vôtres
		Ps2.pa.	2	vôtres
		Ps2.pd.	1	vôtres
		Ps2.s-.	11	vôtre
		Ps2.sa.	3	vôtre
		Ps2.sd.	3	vôtre
		Ps2.sn.	3	vôtre
		Ps2fp-s	8	siennes
		Ps2fpas	2	siennes
		Ps2fpds	5	siennes
		Ps2fpns	2	siennes
		Ps2fs-s	2	tienne
		Ps2fsas	4	tienne
		Ps2fsns	3	tienne
		Ps2mp-s	20	siens
		Ps2mpas	19	siens
		Ps2mpds	16	siens
		Ps2mpns	5	siens
		Ps2ms-s	3	tien
		Ps2msds	1	tiens
		Ps3.p-p	13	leurs
		Ps3.pap	11	leurs
		Ps3.pdp	3	leurs
		Ps3.pnp	5	leurs
		Ps3.s-p	16	leur
		Ps3.sap	14	leur
		Ps3.sdp	9	leur
		Ps3.snp	3	leur
		Ps3fs-s	16	sienne
		Ps3fsas	29	sienne
		Ps3fsds	18	sienne
		Ps3fsns	6	sienne
		Ps3ms-s	13	sien
		Ps3msas	26	sien
		Ps3msds	7	sien
		Ps3msns	2	sien,
PROPE	PPER1P	Pp1.p-	713	nous mêmes
		Pp1.pa	1028	nous mêmes
		Pp1.pd	2250	nous mêmes
		Pp1.pn	6181	nous mêmes
	PPER1S	Pp1.pd	3	même nous
		Pp1.pn	3	même nous
		Pp1.s-	2949	je moi me même
		Pp1.sa	2062	me je moi même ça
		Pp1.sd	4374	je me moi même ça
		Pp1.sn	16 317	je me moi même
		Pp1fpn	2	je
		Pp1fsa	2	moi
		Pp1fsn	2	je

Suite sur la page suivante...

Suite de la page précédente...

smallems	**ems**	**tgb**	**fréq**	**Exemples**
	PPER2P	Pp2.p-	597	vous mêmes
		Pp2.pa	780	vous
		Pp2.pd	1192	vous mêmes
		Pp2.pn	3636	vous mêmes
	PPER2S	Pp2.p-	30	même vous
		Pp2.pa	15	vous même
		Pp2.pd	18	même vous
		Pp2.pn	2	tu
		Pp2.s-	792	te toi tu même
		Pp2.sa	613	te toi même tu
		Pp2.sd	1119	te toi même tu
		Pp2.sn	3272	te tu toi
		Pp2fpn	2	tu
	PPER3P	Pp3.p-	901	les leur
		Pp3.pa	2843	les leur
		Pp3.pd	2101	leur les
		Pp3.pn	26	les leur
		Pp3fp-	586	elles mêmes
		Pp3fpa	42	elles mêmes
		Pp3fpd	663	elles mêmes
		Pp3fpn	3290	elles ils mêmes
		Pp3mp-	1174	eux ils mêmes
		Pp3mpa	164	eux ils mêmes
		Pp3mpd	1975	eux ils mêmes
		Pp3mpn	11 003	eux ils mêmes
	PPER3S	Pp1fs-	11	la
		Pp1fsa	3	la
		Pp1fsd	1	la
		Pp3..-	5321	en y
		Pp3..a	385	en y
		Pp3..d	9601	en y
		Pp3..n	65	en y
		Pp3.s-	2854	lui on même soi
		Pp3.sa	7869	lui on même soi
		Pp3.sd	7754	lui on même seule soi
		Pp3.sn	17 403	lui on même soi
		Pp3fp-	4	elle
		Pp3fpn	4	elle
		Pp3fs-	5873	elle la même
		Pp3fsa	1817	elle la même
		Pp3fsd	1984	elle la lui même
		Pp3fsn	21 099	elle la il même
		Pp3mp-	4	il
		Pp3mpn	4	il
		Pp3ms-	3301	il le lui même
		Pp3msa	4763	il le lui même
		Pp3msd	2197	il le lui même
		Pp3msn	51 184	il le lui même
		Px3..-	46 828	se
		Px3..a	4500	se

Suite sur la page suivante...

Suite de la page précédente...

smallems	ems	tgb	fréq	Exemples
		Px3..d	7535	se
		Px3..n	147	se
PREP	PREP	Sp	825 835	a avec afin après auprès avant chez
PRORE	PRFS	Pr-fs-	988	laquelle
		Pr-fsa	8	laquelle
		Pr-fsd	437	laquelle
		Pr-fsn	107	laquelle
		Ptr-fs-	9	laquelle
		Ptr-fsd	12	laquelle
	PRI	Pr-..-	36 708	dont où que qui quoi
		Pr-..a	8117	où qui dont que quoi
		Pr-..d	2046	quoi dont où que qui
		Pr-..n	35 015	où qui que dont quoi
		Pr-.s-	22	quiconque
		Pr-.sa	2	quiconque
		Pr-.sd	9	quiconque
		Pr-.sn	5	quiconque
		Pr-fp-	659	lesquelles auxquelles de desquelles
		Pr-fpa	47	auxquelles desquelles lesquelles
		Pr-fpd	109	auxquelles desquelles lesquelles
		Pr-fpn	10	auxquelles lesquelles
		Pr-mp-	662	auxquels lesquels desquels
		Pr-mpa	58	auxquels desquels lesquels
		Pr-mpd	111	auxquels desquels lesquels
		Pr-mpn	15	auxquels lesquels
		Ptr-..-	5182	où que qui dont quoi
		Ptr-..a	26	que dont où qui quoi
		Ptr-..d	41	dont où que qui
		Ptr-..n	11	dont où que qui
		Ptr-.s-	13	quiconque
		Ptr-.sa	2	quiconque
		Ptr-fp-	107	auxquelles desquelles lesquelles
		Ptr-fpa	2	auxquelles lesquelles
		Ptr-mp-	147	auxquels desquels lesquels
		Ptr-mpd	4	lesquels
	PRMS	Pr-..-	41	où que quoi
		Pr-..a	1	où
		Pr-..d	15	où quoi
		Pr-..n	35	où que qui quoi
		Pr-fp-	4	lesquelles
		Pr-fpd	1	lesquelles
		Pr-fs-	3	laquelle
		Pr-fsd	5	laquelle
		Pr-mp-	2	lesquels
		Pr-ms-	1011	auquel lequel duquel
		Pr-msa	66	auquel duquel lequel
		Pr-msd	190	auquel duquel lequel
		Pr-msn	188	lequel auquel duquel
		Ptr-..-	92	où que quoi
		Ptr-fp-	1	auxquelles

Suite sur la page suivante...

Suite de la page précédente...

smallems	**ems**	**tgb**	**fréq**	**Exemples**
		Ptr-mp-	6	auxquels desquels
		Ptr-ms-	169	lequel auquel duquel
		Ptr-msd	6	lequel
SUB	SUB	Cs	97 674	afin comme cependant comment lorsque
VCON	VCONP1P	Vacc1p	33	aurions serions
		Vmcc1p	224	irions serions verrions accepterions
	VCONP1S	Vacc1s	139	aurais serais
		Vmcc1s	540	aurais devrais abîmerais accepterais
	VCONP2P	Vacc2p	27	auriez seriez
		Vmcc2p	106	coopéreriez iriez pourriez seriez voudriez
	VCONP2S	Vacc2s	11	aurais serais
		Vmcc2s	103	aimerais aurais oserais prendrais serais
	VCONP3P	Vacc3p	517	seraient auraient seraient
		Vmcc3p	2058	admettraient abandonneraient
	VCONP3S	Vacc3s	1436	aurait serait
		Vmcc3s	9176	aurait bénéficierait dirait pourrait
	VIMPP1P	Vafé1p	2	soyons sommes
		Vmfé1p	1310	abattons abordons acceptons
	VIMPP2P	Vafé2p	15	ayez soyez
		Vmfé2p	1844	abordez acceptez achetez achevez
	VIMPP2S	Vafé2s	9	aie sois
		Vmfé2s	2488	accorde admets aide aie aime amuse
	VINDF1P	Vaif1p	27	aurons serons
		Vmif1p	918	aurons descendrons ferons irons
	VINDF1S	Vaif1s	37	aurai serai
		Vmif1s	953	trouverai tuerai verrai abjurerai
	VINDF2P	Vaif2p	16	aurez serez
		Vmif2p	386	aurez ferez irez nierez prendrez
	VINDF2S	Vaif2s	26	auras seras
		Vmif2s	398	attendras chanteras croiras diras
	VINDF3P	Vaif3p	759	seront auront
		Vmif3p	2469	arrêteront auront donneront feront
	VINDF3S	Vaif3s	1358	sera aura
		Vmif3s	7952	apprendra associera aura baisera
	VINDI1P	Vaii1p	74	avions étions
		Vmii1p	396	abordions accomplissions achetions
	VINDI1S	Vaii1s	253	avais étais
		Vmii1s	1039	allais abordais aboutissais acceptais
	VINDI2P	Vaii2p	32	aviez étiez
		Vmii2p	137	galbiez aimiez alliez approuviez aviez
	VINDI2S	Vaii2s	14	avais étais
		Vmii2s	127	allais avais portais saurais savais
	VINDI3P	Vaii3p	2033	avaient avaient étaient étaient
		Vmii3p	8688	allaient chantaient manquaient
	VINDI3S	Vaii3s	6578	avait était
		Vmii3s	29 396	allait approchait avait buvait cachait
	VINDP1P	Vaip1p	1118	avons sommes
		Vmip1p	2951	avons pouvons saurons savons sommes
	VINDP1S	Vaip1s	2027	ai suis
		Vmip1s	7490	ai dois puis suis sais abaisse abats

Suite sur la page suivante...

Suite de la page précédente...

smallems	ems	tgb	fréq	Exemples
	VINDP2P	Vaip2p	409	avez avez êtes
		Vmip2p	1899	avez avez pouvez savez voulez abordez
	VINDP2S	Vaip2s	256	as es
		Vmip2s	1706	admets apportes as cites comprends
	VINDP3P	Vaip3p	15 173	ont sont
		Vmip3p	46 885	appartiennent arrivent articulent
	VINDP3S	Vaip3s	33 341	a est
		Vmip3s	148 054	a assiste aborde admire agit agite
	VINDPS1P	Vais1p	2	fûmes
		Vmis1p	25	aidâmes arrivâmes donnâmes
	VINDPS1S	Vais1s	19	eus fus
		Vmis1s	342	bardai desservis mirai retransmis
	VINDPS2P	Vais2p	1	fûtes
		Vmis2p	19	connûtes vîtes donnâtes défendîtes
	VINDPS2S	Vmis2s	86	bordas casas feuillas lestas satanas
	VINDPS3P	Vais3p	381	eurent furent
		Vmis3p	3649	abaissèrent abandonnèrent abattirent
	VINDPS3S	Vais3s	1121	eut fut
		Vmis3s	22 997	absorba alla cabana calandra caressa
	VSUBI1P	Vasm1p	1	eussions
		Vmsm1p	2	dussions fussions
	VSUBI1S	Vasm1s	6	eusse
		Vmsm1s	17	avouasse calmasse comprisse coquasse
	VSUBI2P	Vmsm2p	1	pussiez
	VSUBI2S	Vasm2s	1	eusses
		Vmsm2s	9	aimasses coquasses entrasses fisses
	VSUBI3P	Vasm3p	78	eussent fussent
		Vmsm3p	104	allassent connussent consentissent
	VSUBI3S	Vasm3s	455	fût eût fût
		Vmsm3s	617	plût abandonnât acceptât accordât
	VSUBP1P	Vasr1p	14	ayons soyons
		Vmsr1p	48	puissions actions ayons confessions
	VSUBP1S	Vasr1s	23	aie sois
		Vmsr1s	151	matisse admette aie aille amollisse
	VSUBP2P	Vasr2p	4	ayez soyez
		Vmsr2p	28	puissiez ayez fassiez sachiez soyez
	VSUBP2S	Vasr2s	9	aies sois
		Vmsr2s	181	puisses affaires aides aies aiguilles ailles
	VSUBP3P	Vasr3p	710	aient soient
		Vmsr3p	821	manquent accordent aient aillent
	VSUBP3S	Vasr3s	1488	ait soit
		Vmsr3s	3786	aille calme fasse interne jalouse plaise
VINF	VINF	Van–	6603	avoir être
		Vmn–	120 721	abaisser aborder accepter accorder
VPAR	VPARPFP	Vmpapf	15 204	affectées aidées analysées appelées
	VPARPFS	Vmpasf	27 340	absorbée accompagnée accordée
	VPARPMP	Vapapm	5	eus
		Vmpapm	19 116	accoutrés accoutumés accrochés
	VPARPMS	Vapasm	520	eu été
		Vmpasm	69 389	accepté accompagné accueilli accusé

Suite sur la page suivante...

Suite de la page précédente...

smallems	**ems**	**tgb**	**fréq**	**Exemples**
	VPARPRES	Vapp–	2813	ayant étant
		Vmpp–	25 198	abandonnant abordant acceptant
SYMBOLE	SYMBOLE	X	71	x @ \ ] { | ~ #
UEUPH	UEUPH	Ue	14 008	ue l t,
			152 891	! " % & ( ) * + , / # 0 43 5 55 7 1 10
23	128	473	6 468 522	**TOTAUX**

Tableau E.1 – Recensement des étiquettes morphosyntaxiques de tous les mots du corpus avec leur fréquence. La dernière ligne précise le nombre d'étiquettes *smallems*, *ems* et *tgb* observées dans le corpus ainsi que le nombre total de mots. La dernière colonne donne quelques exemples de mots du corpus ayant reçus les trois étiquettes correspondantes sur la ligne. Ces exemples sont constitués par l'étiquette *jeton* des mots mise en minuscule. Certains de ces exemples sont des illustrations d'erreurs d'étiquetage.

E.2 Signification des étiquettes morphosyntaxiques simplifiées

Le tableau E.2 précise la signification des étiquettes morphosyntaxiques simplifiées. Ces étiquettes sont construites en effectuant des regroupements sur les étiquettes morphosyntaxiques *ems*. Lors de ces regroupements, nous avons eu un problème avec l'étiquette *NHMIN*. Cette étiquette, faisant partie des étiquettes non documentées, ne correspond ni aux étiquettes de noms communs commençant par *NC*, ni aux étiquettes de noms propres commençant par *NP*. Nous avons choisi de laisser cette étiquette inchangée. À l'observation des données, il semble que ces étiquettes correspondent à des noms communs masculins invariants en nombre débutant par un h aspiré. Or, cette famille de noms est déjà identifiée par l'étiquette *ems NCHMIN*, peut-être s'agit il d'un dysfonctionnement de l'étiquetage du logiciel Cordial Analyseur.

Étiquette	**Signification**
ADJ	Adjectif
ADV	Adverbe
COO	Conjonction de coordination
DET	Article
INT	Interjection
NCOM	Nom commun
NHMIN	Nom masculin invariant en nombre débutant par un h aspiré
NPRO	Nom propre
PCTFAIB	Ponctuation faible
PCTFORTE	Ponctuation forte
PRODE	Pronom démonstratif
PROIN	Pronom indéfini
PROPO	Pronom possessif
PROPE	Pronom personnel
PREP	Préposition
PRORE	Pronom relatif
SUB	Conjonction de subordination
VCON	Verbe conjugué
VINF	Verbe à l'infinitif
VPAR	Verbe au participe passé ou présent

Tableau E.2 – Signification des étiquettes morphosyntaxiques simplifiées.

E.3 Signification des étiquettes morphosyntaxiques de type Cordial Analyseur

Étiquette	Signification
ADJFP	Adjectif qualificatif féminin pluriel
ADJFS	Adjectif qualificatif féminin singulier
ADJHFS	Adjectif qualificatif féminin singulier débutant par un h aspiré
ADJHMS	Adjectif qualificatif masculin singulier débutant par un h aspiré
ADJHSIG	Adjectif qualificatif invariant en genre singulier débutant par un h aspiré
ADJIND	Adjectif indéfini
ADJINT	Adjectif interrogatif
ADJINV	Adjectif qualificatif invariant en nombre et en genre
ADJMIN	Adjectif qualificatif masculin invariant en nombre
ADJMP	Adjectif qualificatif masculin pluriel
ADJMS	Adjectif qualificatif masculin singulier
ADJNUM	Adjectif numérique cardinal
ADJORD	Adjectif numérique ordinal
ADJPIG	Adjectif qualificatif invariant en genre pluriel
ADJSIG	Adjectif qualificatif invariant en genre singulier
ADV	Adverbe
COO	Conjonction de coordination
DETDEM	Adjectif démonstratif
DETDFS	Article défini féminin singulier
DETDMS	Article défini masculin singulier
DETDPIG	Article défini invariant en genre pluriel
DETIFS	Article indéfini féminin singulier
DETIMS	Article indéfini masculin singulier
DETPOSS	Adjectif possessif
INT	Interjection
NCFIN	Nom commun féminin invariant en nombre
NCFP	Nom commun féminin pluriel
NCFS	Nom commun féminin singulier
NCHFS	Nom commun féminin singulier débutant par un h aspiré
NCHMIN	Nom commun masculin invariant en nombre débutant par un h aspiré
NCHMS	Nom commun masculin singulier débutant par un h aspiré
NCHSIG	Nom commun invariant en genre singulier débutant par un h aspiré
NCI	Nom commun invariant en genre et en nombre
NCMIN	Nom commun masculin invariant en nombre
NCMP	Nom commun masculin pluriel
NCMS	Nom commun masculin singulier
NCPIG	Nom commun invariant en genre pluriel
NCSIG	Nom commun invariant en genre singulier
NHMIN	Nom masculin invariant en nombre débutant par un h aspiré
NPFIN	Nom propre féminin invariant en nombre
NPFP	Nom propre féminin pluriel
NPFS	Nom propre féminin singulier
NPHFS	Nom propre féminin singulier débutant par un h aspiré
NPHMS	Nom propre masculin singulier débutant par un h aspiré
NPHSIG	Nom propre invariant en genre singulier débutant par un h aspiré
NPI	Nom propre invariant en genre et en nombre

Suite sur la page suivante...

Suite de la page précédente...

Étiquette	Signification
NPMIN	Nom propre masculin invariant en nombre
NPMP	Nom propre masculin pluriel
NPMS	Nom propre masculin singulier débutant par un h aspiré
NPPIG	Nom propre invariant en genre pluriel
NPSIG	Nom propre invariant en genre singulier
PCTFAIB	Ponctuation faible
PCTFORTE	Ponctuation forte
PDP	Pronom démonstratif pluriel
PDS	Pronom démonstratif singulier
PIFP	Pronom indéfini féminin pluriel
PIFS	Pronom indéfini féminin singulier
PII	Pronom indéfini invariant en genre et en nombre
PIMP	Pronom indéfini masculin pluriel
PIMS	Pronom indéfini masculin singulier
PIPIG	Pronom indéfini invariant en genre pluriel
PISIG	Pronom indéfini invariant en genre singulier
PP	Pronom possessif
PPER1P	Pronom personnel 1ère personne du pluriel
PPER1S	Pronom personnel 1ère personne du singulier
PPER2P	Pronom personnel 2ème personne du pluriel
PPER2S	Pronom personnel 2ème personne du singulier
PPER3P	Pronom personnel 3ème personne du pluriel
PPER3S	Pronom personnel 3ème personne du singulier
PREP	Préposition
PRFS	Pronom relatif féminin singulier
PRI	Pronom relatif invariant en genre et en nombre
PRMS	Pronom relatif masculin singulier
SUB	Conjonction de subordination
VCONP1P	Verbe indicatif conditionnel 1ère personne du pluriel
VCONP1S	Verbe indicatif conditionnel 1ère personne du singulier
VCONP2P	Verbe indicatif conditionnel 2ème personne du pluriel
VCONP2S	Verbe indicatif conditionnel 2ème personne du singulier
VCONP3P	Verbe indicatif conditionnel 3ème personne du pluriel
VCONP3S	Verbe indicatif conditionnel 3ème personne du singulier
VIMPP1P	Verbe impératif présent 1ère personne du pluriel
VIMPP2P	Verbe impératif présent 2ème personne du pluriel
VIMPP2S	Verbe impératif présent 2ème personne du singulier
VINDF1P	Verbe indicatif futur simple 1ère personne du pluriel
VINDF1S	Verbe indicatif futur simple 1ère personne du singulier
VINDF2P	Verbe indicatif futur simple 2ème personne du pluriel
VINDF2S	Verbe indicatif futur simple 2ème personne du singulier
VINDF3P	Verbe indicatif futur simple 3ème personne du pluriel
VINDF3S	Verbe indicatif futur simple 3ème personne du singulier
VINDI1P	Verbe indicatif imparfait 1ère personne du pluriel
VINDI1S	Verbe indicatif imparfait 1ère personne du singulier
VINDI2P	Verbe indicatif imparfait 2ème personne du pluriel
VINDI2S	Verbe indicatif imparfait 2ème personne du singulier
VINDI3P	Verbe indicatif imparfait 3ème personne du pluriel
VINDI3S	Verbe indicatif imparfait 3ème personne du singulier
VINDP1P	Verbe indicatif présent 1ère personne du pluriel

Suite sur la page suivante...

Suite de la page précédente...

Étiquette	**Signification**
VINDP1S	Verbe indicatif présent 1ère personne du singulier
VINDP2P	Verbe indicatif présent 2ème personne du pluriel
VINDP2S	Verbe indicatif présent 2ème personne du singulier
VINDP3P	Verbe indicatif présent 3ème personne du pluriel
VINDP3S	Verbe indicatif présent 3ème personne du singulier
VINDPS1P	Verbe indicatif passé simple 1ère personne du pluriel
VINDPS1S	Verbe indicatif passé simple 1ère personne du singulier
VINDPS2P	Verbe indicatif passé simple 2ème personne du pluriel
VINDPS2S	Verbe indicatif passé simple 2ème personne du singulier
VINDPS3P	Verbe indicatif passé simple 3ème personne du pluriel
VINDPS3S	Verbe indicatif passé simple 3ème personne du singulier
VINF	Verbe à l'infinitif
VPARPFP	Verbe participe passé féminin pluriel
VPARPFS	Verbe participe passé féminin singulier
VPARPMP	Verbe participe passé masculin pluriel
VPARPMS	Verbe participe passé masculin singulier
VPARPRES	Verbe participe présent
VSUBI1P	Verbe subjonctif imparfait 1ère personne du pluriel
VSUBI1S	Verbe subjonctif imparfait 1ère personne du singulier
VSUBI2P	Verbe subjonctif imparfait 2ème personne du pluriel
VSUBI2S	Verbe subjonctif imparfait 2ème personne du singulier
VSUBI3P	Verbe subjonctif imparfait 3ème personne du pluriel
VSUBI3S	Verbe subjonctif imparfait 3ème personne du singulier
VSUBP1P	Verbe subjonctif présent 1ère personne du pluriel
VSUBP1S	Verbe subjonctif présent 1ère personne du singulier
VSUBP2P	Verbe subjonctif présent 2ème personne du pluriel
VSUBP2S	Verbe subjonctif présent 2ème personne du singulier
VSUBP3P	Verbe subjonctif présent 3ème personne du pluriel
VSUBP3S	Verbe subjonctif présent 3ème personne du singulier

Tableau E.3 – Signification des étiquettes morphosyntaxiques de type CORDIAL ANALYSEUR.

E.4 Signification des étiquettes morphosyntaxiques du projet MULTEXT-GRACE

Étiquette	Signification
Afc??	Adjectif qualificatif comparatif
Af?m?	Adjectif qualificatif masculin
Af?f?	Adjectif qualificatif féminin
Af??s	Adjectif qualificatif singulier
Af??p	Adjectif qualificatif pluriel
Ai-m?-	Adjectif interrogatif masculin
Ai-f?-	Adjectif interrogatif féminin
Ai-?s-	Adjectif interrogatif singulier
Ai-?p-	Adjectif interrogatif pluriel
Ao-m?	Adjectif ordinal masculin
Ao-f?	Adjectif ordinal féminin
Ao-?s	Adjectif ordinal singulier
Ao-?p	Adjectif ordinal pluriel
As-m?	Adjectif possessif masculin
As-f?	Adjectif possessif féminin
As-?s	Adjectif possessif singulier
As-?p	Adjectif possessif pluriel
Cc	Conjonction de coordination
Cs	Conjonction de subordination
Da-m?-d	Article défini masculin
Da-f?-d	Article défini féminin
Da-?s-d	Article défini singulier
Da-?p-d	Article défini pluriel
Da-m?-i	Article indéfini masculin
Da-f?-i	Article indéfini féminin
Da-?s-i	Article indéfini singulier
Da-?p-i	Article indéfini pluriel
Dd-m?-	Adjectif démonstratif masculin
Dd-f?-	Adjectif démonstratif féminin
Dd-?s-	Adjectif démonstratif singulier
Dd-?p-	Adjectif démonstratif pluriel
Ds1???	Adjectif possessif 1ère personne
Ds2???	Adjectif possessif 2ème personne
Ds3???	Adjectif possessif 3ème personne
Ds?m??	Adjectif possessif masculin
Ds?f??	Adjectif possessif féminin
Ds??s?	Adjectif possessif singulier
Ds??p?	Adjectif possessif pluriel
Ds???s	Adjectif possessif au possesseur singulier
Ds???p	Adjectif possessif au possesseur pluriel
Dt-m?-	Adjectif indéfini masculin
Dt-f?-	Adjectif indéfini féminin
Dt-?s-	Adjectif indéfini singulier
Dt-?p-	Adjectif indéfini pluriel

Suite sur la page suivante...

Suite de la page précédente...

Étiquette	Signification
I	Interjection
Mcm?	Adjectif numéral masculin
Mcf?	Adjectif numéral féminin
Mc?s	Adjectif numéral singulier
Mc?p	Adjectif numéral pluriel
Ncm?	Nom commun masculin
Ncf?	Nom commun féminin
Nc?s	Nom commun singulier
Nc?p	Nom commun pluriel
Npm?	Nom propre masculin
Npf?	Nom propre féminin
Np?s	Nom propre singulier
Np?p	Nom propre pluriel
Pd-??n	Pronom démonstratif sujet
Pd-??a	Pronom démonstratif COD
Pd-??d	Pronom démonstratif COI
Pd-m??	Pronom démonstratif masculin
Pd-f??	Pronom démonstratif féminin
Pd-?s?	Pronom démonstratif sigulier
Pd-?p?	Pronom démonstratif pluriel
Pd-??n	Pronom indéfini sujet
Pd-??a	Pronom indéfini COD
Pd-??d	Pronom indéfini COI
Pd-m??	Pronom indéfini masculin
Pd-f??	Pronom indéfini féminin
Pd-?s?	Pronom indéfini sigulier
Pd-?p?	Pronom indéfini pluriel
Pp???n	Pronom personnel sujet
Pp???a	Pronom personnel COD
Pp???d	Pronom personnel COI
Pp1???	Pronom personnel 1ère personne
Pp2???	Pronom personnel 2ème personne
Pp3???	Pronom personnel 3ème personne
Pp?m??	Pronom personnel masculin
Pp?f??	Pronom personnel féminin
Pp??s?	Pronom personnel singulier
Pp??p?	Pronom personnel pluriel
Pr-??n	Pronom relatif sujet
Pr-??a	Pronom relatif COD
Pr-??d	Pronom relatif COI
Pr-m??	Pronom relatif masculin
Pr-f??	Pronom relatif féminin
Pr-?s?	Pronom relatif singulier
Pr-?p?	Pronom relatif pluriel
Ps???n?	Pronom possessif sujet
Ps???a?	Pronom possessif COD
Ps???d?	Pronom possessif COI

Suite sur la page suivante...

Suite de la page précédente...

Étiquette	Signification
Ps1????	Pronom possessif 1ère personne
Ps2????	Pronom possessif 2ème personne
Ps3????	Pronom possessif 3ème personne
Ps????s	Pronom possessif au possesseur singulier
Ps????p	Pronom possessif au possesseur pluriel
Ps?m???	Pronom possessif masculin
Ps?f???	Pronom possessif féminin
Ps??s??	Pronom possessif singulier
Ps??p??	Pronom possessif pluriel
Pt-??n	Pronom interrogatif sujet
Pt-??a	Pronom interrogatif COD
Pt-??d	Pronom interrogatif COI
Pt-m??	Pronom interrogatif masculin
Pt-f??	Pronom interrogatif féminin
Pt-?s?	Pronom interrogatif singulier
Pt-?p?	Pronom interrogatif pluriel
Px-??n	Pronom personnel réfléchi sujet
Px-??a	Pronom personnel réfléchi COD
Px-??d	Pronom personnel réfléchi COI
Px-m??	Pronom personnel réfléchi masculin
Px-f??	Pronom personnel réfléchi féminin
Px-?s?	Pronom personnel réfléchi singulier
Px-?p?	Pronom personnel réfléchi pluriel
Rgc	Adverbe comparatif
Rgp	Adverbe non-comparatif et non de négation
Rgn	Adverbe comparatif de négation
Rpn	Adverbe de négation *ne*
Sp	Préposition
Ypw	Ponctuation de pause
Yps	Ponctuation finale (fin de phrase)
Ypo	Ponctuation de début d'insertion
Ypc	Ponctuation de fin d'insertion
Va????	Verbe auxiliaire
Vm????	Verbe principal
V?n-	Verbe à l'infinitif
V?i???	Verbe à l'indicatif
V?s???	Verbe au subjonctif
V?c???	Verbe au conditionnel
V?f???	Verbe au futur
V?p???	Verbe au participe
V??p??	Verbe au présent
V??i??	Verbe à l'imparfait
V??s??	Verbe au passé
V??f??	Verbe au futur
V??r??	Verbe au subjonctif présent
V??m??	Verbe au subjonctif imparfait
V??c??	Verbe au conditionnel

Suite sur la page suivante...

Suite de la page précédente...

Étiquette	**Signification**
V??é??	Verbe à impératif
V??a??	Verbe au participe passé
V???1?	Verbe conjugué à la 1ère personne
V???2?	Verbe conjugué à la 2ème personne
V???3?	Verbe conjugué à la 3ème personne
V???p?	Verbe au participe pluriel
V???s?	Verbe au participe singulier
V????m	Verbe au participe masculin
V????f	Verbe au participe féminin
V????p	Verbe conjugué au pluriel
V????s	Verbe conjugué au singulier

Tableau E.4 – Signification des étiquettes morphosyntaxiques du projet MULTEXT-GRACE. Dans les étiquettes, le signe ? désigne une place réservée pour d'autres caractères.

E.5 Signification des étiquettes de fonction grammaticale du mot

Étiquette	Signification
A	Constitue l'attribut du sujet
B	Appartient à l'attribut du sujet
C	Constitue le COD
D	Appartient au COD
E	Constitue le COI
F	Appartient au COI
G	Appartient au complément d'agent
H	Circonstanciel
I	Complément infinitif
K	Circonstanciel de temps
L	Circonstanciel de lieu
M	Constitue une apposition
N	Appartient à une apposition
O	Constitue une apostrophe
P	Appartient à une apostrophe
Q	Complément de négation
S	Constitue le SUJET
T	Appartient au SUJET
U	Pronom personnel de pronominalisation
V	Verbe de base de la proposition
Y	Constitue le sujet réel
Z	Appartient au sujet réel
a	Ajout à l'adjectif
d	Reprise du COD
e	Reprise du COI
h	Reprise du circonstanciel
n	Ajout au nom
p	Ajout au pronom
s	Reprise du sujet
t	Ajout au verbe

Tableau E.5 – Signification des étiquettes de fonction grammaticale du mot.

Abréviations et acronymes

ABU : Association des Bibliophiles Universels.
BNC : British National Corpus.
CED : Collins English Dictionaries.
CIDE : Cambridge International Dictionary of English.
DEC : Dictionnaire Explicatif et Combinatoire.
CQP : Corpus Query Processor.
CLS : Cambridge Language Survey.
IGM : Institut Gaspard-Monge.
IR : Information Retrieval.
ITA : Inter-Annotator Agreement.
JOC : Journal Officiel de la Communauté européenne.
LADL : Laboratoire d'Automatique Documentaire et Linguistique
LDOCE : Longman Dictionary Of Contemporary English.
MFC : Microsoft Foundation Class.
MT : Machine Translation.
MVDM : Modified Value Difference Metric.
NLP : Natural Language Processing.
NLU : Natural Language Understanding.
NP : Noun Phrase.
OALDCE : Oxford's Advanced Learner's Dictionary of Current English.
OCR : Optical Character Recognition.
PEBLS : Parallel Exemplar-Based Learning System.
POS : Part-Of-Speech.
SARA : SGML Aware Retrieval Application.
STL : Standard Template Library.
SyntSem : Projet d'étiquetage SYNTaxique et SÉMantique du français.
TAL : Traitement Automatique des Langues.
TALN : Traitement Automatique des Langues Naturelles.
TLF : Trésor de la Langue Française.
VDM : Value Difference Metric.
VP : Verb Phrase.
WSD : Word Sense Disambiguation.

Liste des définitions et théorèmes

Liste des algorithmes

Liste des tableaux

Liste des figures

Bibliographie

Adriaens, G. (1986a). Word expert parsing : A natural language analysis program revised and applied to dutch. *Leuvensche Bijdragen*, *75*(1), 73–154.

Adriaens, G. (1986b). WEP (Word Expert Parsing) revised and applied to dutch. *7th European Conference on Artifical Intelligence (ECAI-1986)*, 222–235.

Adriaens, G. (1989). The parallel expert parser : A meaning-oriented, lexically guided, parallel-interactive model of natural language understanding. *International Workshop on Parsing Technologies*, 309–319.

Adriaens, G. & Small, S. (1988). Word expert revised in a cognitive science perspective. In G. C. S. Small & M. Tanenhaus (Eds.), *Lexical ambiguity resolution : Perspectives from psycholinguistics, neuropsychology, and artificial intelligence* (pp. 13–43). San Mateo, California: Morgan Kaufman.

Agirre, E. & Martinez, D. (2000). Exploring automatic word sense disambiguation with decision lists and the web. *Workshop on Semantic Annotation and Intelligent Content (COLING-2000)*.

Agirre, E. & Martinez, D. (2001a). Knowledge sources for word sense disambiguation. *4th International Conference on Text Speech and Dialogue (TSD-2001)*, 1–10.

Agirre, E. & Martinez, D. (2001b). Learning class-to-class selectional preferences. *5th Computational Natural Language Learning Workshop (ACL-EACL-CoNLL-2001)*, 15–22.

ALPAC. (1966). *Language and machine : Computers in translation and linguistics* (Tech. Rep.). Washington, D.C.: National Research Council Automatic Language Processing Advisory Committee.

Amsler, R. (1980). *The structure of the merriam-webster pocket dictionary.* Ph. d. dissertation, University of Texas at Austin, Austin, Texas. (293 pp.)

Anderson, J. (1976). *Language, memory, and thought.* Hillsdale, New Jersey: Lawrence Erlbaum and Associates.

Anderson, J. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, *22*(3), 261–295.

Atkins, B. & Levin, B. (1991). Admitting impediments. In U. Zernik (Ed.), *Lexical acquisition : Exploiting on-line resources to build a lexicon* (pp. 233–262). Hillsdale, New Jersey: Lawrence Erlbaum and Associates.

Audibert, L. (2002). Etude des critères de désambiguïsation sémantique automatique : Présentation et premiers résultats sur les cooccurrences. *6ème Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL-2002)*, 415–424.

Bar-Hillel, Y. (1960). The present status of automatic translation of languages. In D. Booth & R. E. Meagher (Eds.), *Advances in computers* (Vol. 1, pp. 91–163). New York: Academic Press.

Bel'skaja, I. K. (1957). Machine translation of language. *Reserach*, *10*(10).

Bigi, B. & Smaïli, K. (2002). Identification thématique hiérarchique : Application aux forums de discussions. *9ème conférence sur le Traitement Automatique des Langues Naturelles (TALN-2002)*, *1*, 115–124.

Bishop, Y. M., Fienberg, S. E. & Holland, P. W. (1975). *Discrete multivariate analysis : Theory and practice.* MIT Press. (ISBN : 0262520400)

Black, E. (1988). An experiment in computational discrimination of english word senses. *IBM Journal of Research and Development*, *32*(2), 185–194.

Bloomfield, L. (1933). *Language.* New York: Holt.

Boguraev, B. (1979). *Automatic resolution of word-sense ambiguity.* Unpublished doctoral dissertation, Cambridge University.

Bookman, L. (1987). A microfeature based scheme for modelling semantics. *10th International Joint Conference on Artificial Intelligence (IJCAI-1987)*, 611–614.

Braden-Harder, L. (1993). Sense disambiguation using on-line dictionaries. In *Natural language processing : The PLPNLP approach* (pp. 247–261). Dordrecht: Kluwer Academic Publishers.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984). Classification and regression trees. In *Institute of electrical and electronics engineers (IEEE), transactions on pattern analysis and machine intelligence* (Vol. 5, pp. 179–190). Pacific Grove, California: Wadsworth and Brooks.

Briscoe, E. (1991). Lexical issues in natural language processing. *Natural Language and Speech*, 39–68.

Brown, P., Pietra, S. D., Pietra, V. D. & Mercer, R. (1991). Word-sense disambiguation using statistical methods. *29th Annual Meeting of the Association for Computational Linguistics (ACL-1991)*, 264–270.

Brown, P. F., Della Pietra, V. J., deSouza, P. V., Lai, J. C. & Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, *18*(4), 467–479.

Bruce, R. & Wiebe, J. (1994a). A new approach to word sense disambiguation. *ARPA Workshop on Human Language Technology*, 244–249.

Bruce, R. & Wiebe, J. (1994b). Word-sense disambiguation using decomposable models. *32nd Annual Meeting of the Association for Computational Linguistics (ACL-1994)*, 139–145.

Bruce, R. & Wiebe, J. (1998). Word-sense distinguishability and inter-coder agreement. *3rd Conference on Empirical Methods in Natural Language Processing (EMNLP-1998)*, 53–60.

Bruce, R., Wiebe, J. & Perdersen, T. (1996). The measure of a model. *1st Conference on Empirical Methods in Natural Language Processing (EMNLP-1996)*, 101–112.

Brun, A., Smaïli, K. & Haton, J.-P. (2002). WSIM : une méthode de détectin de thème fondée sur la similarité entre mots. *9ème conférence sur le Traitement Automatique des Langues Naturelles (TALN-2002)*, *1*, 145–154.

Bryan, R. (1973). Abstract thesauri and graph theory applications to thesaurus research. In S. Y. Sedelow (Ed.), *Automated language analysis* (pp. 45–89). Lawrence, Kansas: University of Kansas.

Bryan, R. (1974). Modelling in thesaurus research. In S. Y. Sedelow & al. (Eds.), *Automated language analysis* (pp. 44–59). Lawrence, Kansas: University of Kansas.

Buitelaar, P. (1997). A lexicon for underspecified semantic tagging. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1997) : Workshop « Tagging Text with Lexical Semantics : Why, What, and How ? »*, 25–33.

Byrd, R., Calzolari, N., Chodorov, M., Klavans, J., Neff, M. & Rizk, O. (1987). Tools and methods for computational linguistics. *Computational Linguistics*, *13*(3/4), 219–240.

Calzolari, N. (1984). Detecting patterns in a lexical data base. *10th International Conference on Computational Linguistics (COLING-1984)*, 170–173.

Charniak, E. (2000). A maximum-entropy-inspired parser. *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 132–139.

Chodorow, M., Byrd, R. & Heidorn, G. (1985). Extracting semantic hierarchies from a large on-line dictionary. *23rd Annual Meeting of the Association for Computational Linguistics (ACL-1985)*, 299–304.

Christ, O. (1994). A modular and flexible architecture for an integrated corpus query system. *3rd Conference on Computational Lexicography and Text Research (COMPLEX-1994)*, 23–32.

Collins, A. & Loftus, E. (1975). A spreading activation theory of semantic processing. *Psychological Review*, *82*(6), 407–428.

Cordial Analyseur [Logiciel pour L'étiquetage morpho-syntaxique et la lemmatisation]. (1994–2002). (c) Synapse Développement. (http://www.synapse-fr.com)

Cost, S. & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, *10*, 57–78.

Cottrell, G. (1985). *A connectionist approach to word-sense disambiguation.* Ph. d. dissertation, Department of Computer Science, University of Rochester.

Cottrell, G. & Small, S. (1983). A connectionist scheme for modelling word sense disambiguation. *Cognition and Brain Theory*, *6*, 89–120.

Cowie, J., Guthrie, J. & Guthrie, L. (1992). Lexical disambiguation using simulated annealing. *14th International Conference on Computational Linguistics (COLING-1992)*, 359–365.

Cussens, J. (1993). Bayes and pseudo-bayes estimates of conditional probability and their reliability. *6th European Conference on Machine Learning (ECML-1993)*, 136–152.

Daelemans, W. (1999). Machine learning approches. In H. V. Halteren (Ed.), *Syntactic wordclass tagging* (pp. 285–304). Kluwer Academic Publishers.

Daelemans, W. & Hoste, V. (2002). Evaluation of machine learning methods for natural language processing tasks. *3rd International Conference on Language Resources and Evaluation (LREC-2002)*, 755–760.

Daelemans, W., Hoste, V., Meulder, F. D. & Naudts, B. (2003). Combined optimization of feature selection and algorithm parameter interaction in machine learning of language. *14th European Conference on Machine Learning (ECML-2003)*.

Dagan, I. & Itai, A. (1994). Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics*, *20*(4), 563–596.

Dagan, I., Itai, A. & Schwall, U. (1991). Two languages are more informatice than one. *29th Annual Meeting of the Association for Computational Linguistics (ACL-1991)*, 130–137.

Dagan, I., Marcus, S. & Markovitch, S. (1993). Contextual word similarity and estimation from sparse data. *31st Annual Meeting of the Association for Computational Linguistics (ACL-1993)*, 22–26.

Dahlgren, K. (1988). *Naive semantics for natural language understanding.* Boston: Kluwer Academic Publishers.

Denis, F. & Gilleron, R. (2000, juin). *Apprentissage à partir d'Exemples* [Notes de cours]. Universités de Lille 3. (http://www.grappa.univ-lille3.fr/polys/apprentissage/index.html)

Domingos, P. & Pazzani, M. (1997). Beyond independence : Conditions for the optimality of the simple bayesian classifier. *Machine Learning*, *29*, 103–130.

Dostert, L. E. (1955). The Georgetown I.B.M. experiment. In W. N. Locke & A. D. Booth (Eds.), *Machine translation of languages* (pp. 124–135). New York: John Wiley and Sons.

Eaton, H. S. (1940). *Semantic frequency list for English, French, German and Spanish.* Chicago University. Chicago.

El-Bèze, M., Loupy, C. de & Marteau, P.-F. (1998). WSD based on three short context methods. *SENSEVAL Workshop.*

El-Bèze, M., Michelon, P. & Pernaud, R. (1999, October). *An integer programming approach to word sense disambiguation.* (Submitted to European Journal of Operational Research (EJOR-1999))

Escudero, G., Marquez, L. & Rigau, G. (2000a). Boosting applied to word sense disambiguation. *11th European Conference on Machine Learning (ECML-2000)*, 129-141.

Escudero, G., Marquez, L. & Rigau, G. (2000b). A comparison between supervised learning algorithms for word sense disambiguation. *4th Computational Natural Language Learning Workshop (CoNLL-2000).*

Escudero, G., Marquez, L. & Rigau, G. (2000c). Naive bayes and exemplar-based approaches to word sense disambiguation revisited. *14th European Conference on Artifical Intelligence (ECAI-2000).*

Feldman, J. & Ballard, D. (1982). Connectionist models and their properties. *Cognitive Science*, *6*(3), 205–254.

Fellbaum, C. (Ed.). (1998). *Wordnet : An electronic lexical database.* Cambridge, Massachusetts: MIT Press. (ISBN 0-262-06197-X)

Fellbaum, C. (1999). *The organisation of verbs and verb concepts in a semantic net.* Dordrecht: Kluwer Academic Publishers.

Ferret, O. (2002). Segmenter et structurer thématiquement des textes par l'Utilisation conjointe de collocations et de la récurrence lexicale. *9ème conférence sur le Traitement Automatique des Langues Naturelles (TALN-2002)*, *1*, 155–164.

Ferret, O., Grau, B., Minel, J. L. & Porhiel, S. (2001). Repérage de structures thématiques dans des textes. *8ème conférence sur le Traitement Automatique des Langues Naturelles (TALN-2001)*, *1*, 163–172.

Fontenelle, T. (1990). Automatic extraction of lexical-semantic relations from dictionary definitions. *4th International Congress on Lexicography (EURALEX-1990)*, 89–103.

Francois, J., Victorri, B. & Manguin, J.-L. (1999). Polysémie adjectivale et synonymie. *Colloque POLYSEMIE.*

Gale, W., Church, K. & Yarowsky, D. (1992a). Estimating upper and lower bounds on the performance of word-sense disambiguation programs. *30th Annual Meeting of the Association for Computational Linguistics (ACL-1992)*, 249–256.

Gale, W., Church, K. & Yarowsky, D. (1992b). A method for disambiguating word senses in a large corpus. In *Computers and the humanities* (Vol. 26, pp. 415–439). Kluwer Academic Publishers.

Gale, W., Church, K. & Yarowsky, D. (1992c). One sense per discourse. *Speech and Natural Language Workshop*, 233–237.

Gale, W., Church, K. & Yarowsky, D. (1992d). Using bilingual materials to develop word sense disambiguation methods. *International Conference on Theoretical and Methodological Issues in Machine Translation*, 101–112.

Golding, A. (1995). A bayesion hybrid method for context-sensitive spelling correction. *3th Workshop on Very Large Corpora*, 39–53.

Gould, R. (1957). Multiple correspondance. *Mechanical translation*, *4*(1/2), 14–27.

Grishman, R., MacLeod, C. & Meyers, A. (1994). COMLEX syntax : Building a computational lexicon. *15th International Conference on Computational Linguistics (COLING-1994)*, 268–272.

Grishman, R., MacLeod, C. & Meyers, A. (1999). A large syntactic dictionary for natural language processing. *Computers and the Humanities*.

Grishman, R. & Sterling, J. (1993). Smoothing of automatically generated selectional constraints. In *Human language technology* (pp. 254–259). Morgan Kaufmann.

Gross, G. & Clas, A. (1997). Synonymie, polysémie et classes d'objets. In (Vol. 42, pp. 147–155). Presses de l'Université de Montréal.

Guthrie, J., Guthrie, L., Wilks, Y. & Aidinejad, H. (1991). Subject-dependent co-occurrence and word sense disambiguation. *29th Annual Meeting of the Association for Computational Linguistics (ACL-1991)*, 146–152.

Harley, A. & Glennon, D. (1997). Sense tagging in action : Combining different test with additive weighings. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1997) : Workshop « Tagging Text with Lexical Semantics : Why, What, and How ? »*, 74–78.

Harper, K. E. (1957a). Contextual analysis. *Mechanical Translation*, *4*(3), 70–75.

Harper, K. E. (1957b). Semantic ambiguity. *Mechanical Translation*, *4*(3), 68–69.

Harris, S. Z. (1954). Distributional structure. *Word*, *10*, 146–162.

Hayes, J. (1976). *A process to implement some word-sense disambiguation* (Working paper No. 23). Université de Genève: Institut pour les études Sémantiques et Cognitives de Genève.

Hayes, P. (1977a). On semantic nets, frames and associations. *5th International Joint Conference on Artificial Intelligence*, 99–107.

Hayes, P. (1977b). *Some association-based techniques for lexical disambiguation by machine.* Doctoral dissertation, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, Lausanne.

Hayes, P. (1978). *Mapping input into schemas* (Technical report No. 29). Departement of Computeur Science, University of Rochester.

Hearst, M. (1991). Noun homograph disambiguation using local context in large text corpora. *7th Annual Conference of the University of Waterloo Centre for the New OED and Text Research*, 1–22.

Hiro, K., Wu, H. & Furugori, T. (1996). Word-sense disambiguation with a corpus-based semantic network. *Journal of Quantitative Linguistics*, *3*(3), 244–251.

Hirst, G. (1987). Semantic interpretation and the resolution of ambiguity. *Studies in Natural Language Processing*.

Hjemslev, L. (1953). *Prolegomena to theory of language.* Bloomington, Indiana. (Translated from Danish)

Hobbs, J. (1987). World knowledge and word meaning. *3rd Workshop on Theoretical Issues in Natural Language Processing (TINLAP-3)*, 20–25.

Holland, G. (1962). Outline for a logical theory of adaptative systems. *Journal for the Association of Computing Machinery*, *3*.

Ide, N. & Véronis, J. (1991). An assessment of information automatically extracted from machine readable dictionaries. *5th Conference of the European Chapter of the Association for Computational Linguistics (EACL-1991)*, 227–232.

Ide, N. & Véronis, J. (1993a). Knowledge extraction from machine-readable dictionaries : An evaluation. *3rd International European Association for Machine Translation (EAMT) Workchop « Machine Translation and the Lexicon »*.

Ide, N. & Véronis, J. (1993b). Refining taxonomies extracted from machine-readable dictionaries. *Research in Humanities Computing II*, 145–159.

Ide, N. & Véronis, J. (1998). Word sense disambiguation : The state of the art. *Computational Linguistics : Special Issue on Word Sense Disambiguation*, *24*, 1–40.

Imbs, P. (1971). *Trésor de la langue française. dictionnaire de la langue du XIXème et du XXème siècle (1989-1960).* Editions du Centre National de la Recherche Scientifique. Paris.

Janssen, S. (1992). Tracing cohesive relations in corpora samples using dictionary data. *New Directions in English Language Corpora : Methodology, Results, Software Develpoments*, 143–152.

Johansson, S. (1980). The LOB corpus of british english texts : Presentation and comments. *Association for Literary and Linguistic Computing Journal (ALLC-1980)*, *1*(1), 25–36.

Jorgensen, J. (1990). The psychological reality of word senses. *Journal of psycholinguistic research*, *19*, 167–190.

Kaplan, A. (1955). An experimental study of ambiguity and context. *Mechanisme Translation*, *2*, 39–46. (Première publication : Mimeographed, November 1950)

Katz, J. & Fodor, J. (1964). The structure of a semantic theory. *The Structure of Language*, 479–518.

Kawamoto, A. (1988). Distributed representations of ambiguous words and their resolution in a connectionist network. In S. S. . G. C. . M. Tanenhaus (Ed.), *Lexical ambiguity resolution : Perspectives from psycholinguistics, neuropsychology, and artifical intelligence* (pp. 195–228). San Mateo, California: Morgan Kaufman.

Kelly, E. F. & Stone, P. J. (1975). *Computer recognition of english word senses.* North-Holland Publishing. North-Holland, Amsterdam.

Ketterlin, A. (2001). *Apprentissage symbolique supervisé* [Notes de cours]. Universités de Lille 3. (plus disponible en ligne)

Kilgarrif, A. & Rosenzweig, J. (2000). English SENSEVAL : Report and results. *2nd International Conference on Language Resources and Evaluation (LREC-2000)*, *3*, 1239–1244. (http://www.itri.brighton.ac.uk/events/senseval/athens.ps)

Kilgarriff, A. (1992). *Polysemy.* Unpublished doctoral dissertation, University of Sussex, United Kingdom.

Kilgarriff, A. (1994). The myth of completness and some problems with consistency (the role of frequency in deciding what goes in the dictionary). *6th International Congress on Lexicography (EURALEX-1994)*, 101–106.

Kilgarriff, A. (1997a). Evaluating word sense disambiguation programs : Progress report. *Speech and Language Technology (SALT-1997) Workshop on Evaluation in Speech and Language Technology*, 114–120.

Kilgarriff, A. (1997b). What is word sense disambiguation good for. *4th Natural Language Processing Pacific Rim Symposium (NLPRS-1997)*, 209–214.

Kilgarriff, A. (1998a). Gold standard datasets for evaluatinf word sense disambiguation programs. *Computer Speech and Language*, *12*(3), 453–472.

Kilgarriff, A. (1998b). SENSEVAL : An exercise in evaluating word sense disambiguation programs. *8th International Congress on Lexicography (EURALEX-1998)*, 176–174.

Klavans, J., Chodorow, M. & Wacholder, N. (1990). From dictionary to knowledge base via taxonomy. *6th Conference of the UW Centre for the New OED*, 110–132.

Krovetz, R. (1998). More than one sense per discourse. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1998) Workshop.*

Krovetz, R. & Croft, W. (1989). Word sense disambiguation using machine-readable dictionaries. *Association for Computing Machinery Special Interest Group on Information Retrieval (ACM-SIGIR-1989) : 12th Annual International Conference on Research and Development in Information Retrieval*, 127–136.

Krovetz, R. & Croft, W. (1992). Lexical ambiguity and information retrieval. *Transactions on Information Systems (TOIS) - Publication of the Association for Computing Machinery (ACM)*, *10*(2), 115–141.

Kucera, H. & Francis, W. N. (1967). *Computational analysis of present-day american english.* Brown University. Providence.

Leacock, C., Chodorow, M. & Miller, G. (1998). Using corpus statistics and WordNet relations for sense identification. *Computers and the Humanities*, *31*, 147–165.

Lee, J., Kim, M. & Lee, Y. (1993). Information retrieval based on conceptual distance in IS-a hierrarchies. *Journal of Documentation*, *49*(2), 188–207.

Lenat, D. B. & Guha, R. (1989). *Building large knowledge-based systems : Representation and inference in the cyc project.* Massachusetts: Addison-Wesley. (ISBN : 0201517523)

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries : How to tell a pine cone from an ice cream cone. *Special Interest Group for DOCumentation (SIGDOC-1986)*, *17*(4), 24–26.

Liddy, E. & Paik, W. (1993). Statistically-guided word sense disambiguation. *American Association for Artificial Intelligence (AAAI) Fall Symposium*, 98–107.

Lorge, I. (1949). *Semantic content of the 470 commonest english words.* Columbia University. New York.

Luk, A. K. (1995). Statistical sense disambiguation with relatively small corpora using dictionary definitions. *33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, 181–188.

Luk, A. K. (1996). *A conceptula cohesion based approach to word sense disambiguation.* Unpublished doctoral dissertation, Department of Computing, Macquarie Univversity, Macquarie University NSW 2109, Australia.

Mahesh, K., Nirenburg, S. & Beale, S. (1997). If you have it, flaunt it : Using full ontological knowledge for word sense disambiguation. *7th International Conference on Theoretical and Methodological Issues in Machine Translation*, 1–9.

Mahesh, K., Nirenburg, S., Beale, S., Viegas, E., Raskin, V. & Onyshkevych, B. (1997). Word sense disambiguation : Why statistics when we have these numbers. *7th International Conference on Theoretical and Methodological Issues in Machine Translation*, 151–159.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz k. & Schasberger, B. (1994). The penn treebank : Annotating predicate argument structure. *Human Language Technology Workshop.*

Marcus, M. & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english : The penn treebank. *Computational Linguistics*, *19*.

Markowitz, J., Ahlswede, T. & Evens, M. (1986). Semantically significant patterns in dictionary definitions. *24th Annual Meeting of the Association for Computational Linguistics (ACL-1986)*, 112–119.

Masterman, M. (1957). The thesaurus in syntax and semantics. *Mechanical Translation*, *4*, 1–2.

Masterman, M. (1961). Semantic message detection for machine translation, using an interlangua. *International Conference on Machine Translation of Languages and Applied Language Analysis*, 437–475.

McClelland, J. & Rumelhart, D. (1981). An interactive activation of context effects in letter perception : Part 1. an account of basic findings. *Psychological Review*, *88*, 115–133.

McRoy, S. (1992). Using multiple knowledge sources for word sense discrimination. *Computational Linguistics*, *18*, 1–30.

Meillet, A. (1926). *Linguistique historique et linguistique générale* (Vol. 1, 2 ed.). Paris: Champion.

Mel'cuk, I., Clas, A. & Polguere, A. (1995). *Introduction à la lexicologie explicative et combinatoire.* Louvain-la-Neuve, AUPELF-UREF/Duculot.

Meyer, D. & Schvaneveldt, R. (1971). Facilitation in recognizing pairs of words : Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, *90*(2), 227–234.

Michiels, A. (1982). *Exploiting a large dictionary data base.* Doctoral dissertation, Université de Liège, Liège, Belgique.

Michiels, A., Mullenders, J. & Noël, J. (1980). Exploiting a large database by longman. *8th International Conference on Computational Linguistics (COLING-1980)*, 374–382.

Mihalcea, R. & Moldovan, D. (1998). Word sense disambiguation based on semantic density. *Workshop on Usage of Wordnet Natural Language Processing Systems (COLING-ACL-1998).*

Miller, G., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. (1990). WordNet : An on-line lexical database. *International Journal of Lexicography*, *3*(4), 235–244.

Mitchell, T. M. (1997). *Machine learning, McGraw-hill series in computer science.* WCB McGraw-Hill. (ISBN : 0070428077)

Mooney, R. (1996). Comparative experiments on disambiguating word senses : An illustration of the role of bias in machine learning. *1st Conference on Empirical Methods in Natural Language Processing (EMNLP-1996)*, 82–91.

Nakamura, J. & Nagao, M. (1988). Extraction of semantic information from an ordinary english dictionary and its evaluation. *12th International Conference on Computational Linguistics (COLING-1988)*, 459–464.

Ng, H. (1997). Exemplar-based word sense disambiguation : Some recent improvements. *2nd Conference on Empirical Methods in Natural Language Processing (EMNLP-1997)*, 208–213.

Ng, H. T. (1997). Getting serious about word sense disambiguation. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1997) : Workshop « Tagging Text with Lexical Semantics : Why, What, and How? »*, 1–7. (Invited paper)

Ng, H. T. & Lee, Y. K. (1996). Integrating multiple knowledge sources to disambiguate word sense : An exemplar-based approach. *34th Annual Meeting of the Society for Computational Linguistics*(17(4)), 40–47.

Ng, H. T. & Lee, Y. K. (2002). An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. *7th Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, 41–48.

Ng, H. T., Lim, C. Y. & Foo, S. K. (1999). A case study on inter-annotator agreement for word sense disambiguation. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1999) Workshop on Standardizing Lexical Resources*, 9–13.

Ng, H. T. & Zelle, J. (1997). Corpus-based approaches to semantic interpretation in natural language processing. *Artificial Intelligence Magazine - Special Issue on Natural Language Processing*, *18*(4), 45–64.

Oettinger, A. G. (1955). The design of an automatic russian-english technical dictionary. In W. N. Locke & A. D. Booth (Eds.), *Machine translation of languages* (pp. 47–65). New York: John Wiley and Sons.

Oswald, V. A. J. (1952). Microsemantics. *M.I.T. Conference on Mechanical Translation*, Mimeographed, 10 pp.

Oswald, V. A. J. (1957). The rationale of idioglossary technique. In L. E. Dostert (Ed.), *Research in machine translation* (pp. 63–69). Washington, D.C.: Georgetown University Press.

Oswald, V. A. J. & Lawson, R. H. (1953). An idioglossary for mechanical translation. *Moderne Language Forum*, *38*(3/4), 1–11.

Palmer, H. (1933). *2nd interim report on english collocations.* Institute for Research in English Teaching. Tokyo.

Palmer, M. (1998). Are WordNet sense distinctions appropriate for computational lexicons. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1998) : SENSEVAL.*

Panov, D. (1960). La traduction mécanique et l'Humanité. *Impact*, *10*(1), 17–25.

Parker-Rhodes, A. F. (1958). The use of statistics in language research. *Mechanical Translation*, *5*(2), 67–73.

Patrick, A. (1985). *An exploration of abstract thesaurus instantiation.* M. sc. thesis, University of Kansas, Lawrence, Kansas.

Pedersen, T. (2001). Machine learning with lexical features : The duluth approach to senseval-2. *2nd International Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*, 139–142.

Pedersen, T. (2002). A baseline methodology for word sense disambiguation. *Conferences on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, 126–135.

Pedersen, T. & Bruce, R. (1997a). Distinguishing word senses in untagged text. *2nd Conference on Empirical Methods in Natural Language Processing (EMNLP-1997).*

Pedersen, T. & Bruce, R. (1997b). A new supervised learning algorithm for word sense disambiguation. *14th National Conference on Artificial Intelligence (AI-1997).*

Pedersen, T., Bruce, R. & Wiebe, J. (1997). Sequential model selection for word sense disambiguation. *5th Conference on Applied Natural Language Processing (ANLP-1997).*

Pereira, F. & Tischby, N. (1992). Distributional similarity, phase transitions and hierarchical clustering. *Working Notes of the American Association for Artificial Intelligence (AAAI) Symposium on Probabilistic Approaches to Natural Language*, 108–112.

Pereira, F., Tishby, N. & Lee, L. (1993). Distributional clustering of English words. *31st Annual Meeting of the Association for Computational Linguistics (ACL-1993)*, 183-190.

Pimsleur, P. (1957). Semantic frequency counts. *Mechanical Translation*, *4*(1–2), 11–13.

Procter, P. (Ed.). (1995). *Cambridge international dictionary of english.* Cambridge university press. Cambridge.

Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics*, *17*(4), 409–441.

Pustejovsky, J., Boguraev, B. & Johnston, M. (1995). *A core lexical engine : The contextual determination of word sense* (Tech. Rep.). Department of Computer Science, Brandeis University.

Quillian, M. (1961). A design for an understanding machine. *Communication Presented at the Colloquium Semantic Problems in Natural Language.*

Quillian, M. (1962a). A revised design for an understanding machine. *Mechanical Translation*, *7*(1), 17–29.

Quillian, M. (1962b). *A semantic coding technique for machanical english paraphrasing* (Tech. Rep.). Research Laboratory of Electronics, M.I.T.: International memorandum of the Mechanical translation Group.

Quillian, M. (1967). Word concepts : A theory and simulation of some basic semantic capabilities. *Behavioral Science*, *12*, 410–430.

Quillian, M. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic information processing* (pp. 227–270). MIT Press.

Quillian, M. (1969). The teachable language comprehender : A simulation program and theory of language. *Communications of the Association for Computing Machinery (ACM-1969)*, *12*(8), 459–476.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. (1993). *C4.5 programs for machine learning* (1 ed.). Morgan Kaufmann. (ISBN : 1558602380)

Rada, R., Mili, H., Bicknell, E. & Blettner, M. (1989). Development and application of a metric on semantic nets. *Institute of Electrical and Electronics Engineers (IEEE) : Transaction on Systems, Man, and Cybernetics*, *19*(1), 17–30.

Reifler, E. (1955). The mechanical determination of meaning. In N. W. Locke & A. D. Booth (Eds.), *Machine translation of languages* (pp. 136–164). New York: John Wiley and Sons.

Resnik, P. (1992). WorNet and distributional analysis : A class-based approach to statistical discovery. *American Association for Artificial Intelligence (AAAI) Workshop on Statistically-Based Natural Language Processing Techniques*, 48–56.

Resnik, P. (1993a). *Selection and information : A class-based approach to lexical relationships.* Unpublished doctoral dissertation, University of Pennsylvania.

Resnik, P. (1993b). Semantic classes and syntactic ambiguity. *ARPA Workshop on Human Language Technology*, 278–283.

Resnik, P. (1995). Disambiguating noun groupings with repect to WordNet senses. *3th Workshop on Very Large Corpora*, 54–68.

Resnik, P. (1997). Selectional preference and sense disambiguation. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1997) : Workshop « Tagging Text with Lexical Semantics : Why, What, and How ? »*, 95–130.

Resnik, P. & Yarowsky, D. (1997). A perspective on word sense disambiguation methods and their evaluation. *Association for Computational Linguistics Special Interest Group on the Lexicon (ACL-SIGLEX-1997) : Workshop « Tagging Text with Lexical Semantics : Why, What, and How ? »*, 79–86.

Resnik, P. & Yarowsky, D. (2000). Distinguishing systems and distinguishing senses : New evaluation methods for word sense disambiguation. In *Natural language engineering* (Vol. 5, pp. 113–133). Cambridge university press.

Reymond, D. (2001). Dictionnaires distributionnels et étiquetage lexical de corpus. *5ème Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL-2002)*, *1*, 479–488.

Reymond, D. (2002). Méthodologie pour la création d'un dictionnaire distributionnel dans une perspective d'étiquetage lexical semi-automatique. *6ème Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL-2002)*, *1*, 405–414.

Richards, I. A. (1953). Towards a theory of translation. *Studies in Chinese Thought*.

Richardson, R. & Smeaton, A. (1994). *Automatic word sense disambiguation in KBIR application.* Working paper CA-0595, School of Computer Applications, Dublin City University. Dublin, Ireland.

Richens, R. (1958). Interlingual machine translation. *Computer Journal*, *1*(3), 144–147.

Rijsbergen, C. V. (1979). *Information retrieval* (2 ed.). London: Butterworths.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2*, 229–246.

Salton, G., Wong, A. & Yang, C. S. (1975). A vector space for information retrieval. *Communication of the Association for Computing Machinery (ACM)*, *18*(11), 613–620.

Sanderson, M. (1994). Word sens disembiguation and information retrieval. *Association for Computing Machinery Special Interest Group on Information Retrieval (ACM-SIGIR-1994) : 17th Annual International Conference on Research and Development in Information Retrieval*, 142–151.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. *International Conference on New Methods in Language Processing*.

Schütze, H. (1992). Dimensions of meaning. *Supercomputing-1992*, 787–796.

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics : Special Issue on Word Sense Disambiguation*, *24*(1), 97–123.

Schütze, H. & Pedersen, J. (1995). Information retrieval based on word senses. *Symposium on Document Analysis and Information Retrieval (SDAIR-1995)*.

Sedelow, S. Y. & Donna, D. W. (1988). Knowledge retrieval from domain-transcendent expert systems : II. research results. *Annual Meeting of American Society for Information Science (ASIS)*, 209–212.

Sedelow, S. Y. & Sedelow, W. A. J. (1969). Categories and procedures for content analysis in the humanities. *The Analysis of Communication Content*, 487–499.

Sedelow, S. Y. & Sedelow, W. A. J. (1986). Thesaurus knowledge representation. *University of Waterloo Conference on Lexicology*, 29–43.

Sedelow, S. Y. & Sedelow, W. A. J. (1992). Recent model-based and model-related studies of a large-scale lexical resource (roget's thesaurus). *14th International Conference on Computational Linguistics (COLING-1992)*, 1223–1227.

Sinclair, J. (Ed.). (1987). *Looking up : An account of the COBUILD project in lexical computing.* London: Collins.

Slator, B. (1992). Sense and preference. *Computer and Mathematics with Applications*, *23*(6/9), 391–402.

Small, S. (1980). *Word expert parsing : A theory of distributed word-based natural language understanding.* Doctoral dissertation, Department of Computer Science, University of Marland. (available as technical report 954)

Small, S. (1983). Parsing as cooperative distributed inference : Understanding through memory interactions. In M. King (Ed.), *Parsing natural language* (pp. 247–276). London: Academic Press.

Small, S. & Rieger, C. (1982). Parsing and comprehencing with word experts (a theory and its realization). In L. Wendy & R. Martin (Eds.), *Strategies for natural language processing* (pp. 89–147). Hillsdale, New Jersey: Lawrence Erlbaum and Associates.

Sparck Jones, K. (1964). *Synonymy and semantic classification.* Ph. d. thesis, University of Cambridge, Cambridge, England.

Sparck Jones, K. (1986). *Synonymy and semantic classification.* Edinburgh, England: Edinburgh University Press.

Stanfill, C. & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the Association for Computing Machinery (ACM)*, *29:12*, 1213–1228.

Stevenson, M. & Wilks, Y. (2001). The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics*, *27*(3), 321–349.

Sussna, M. (1993). Word sense disambiguation for free-text indexing using a massive semantic network. *2nd International Conference on Information and Knowledge Management (CIKM-1993)*, 67–74.

Sutcliffe, R. F. E., McElligott, A., O'Sullivan, D., Polikarpov, A. A., Kuzmin, L. A., Néill, G. O. & Véronis, J. (1996a). An interactive approach to the creation of a multilingual concept ontology for language engineering. *European Conference on Artificial Intelligence (ECAI-1996) : Workshop Multilinguality in the Sofware Industry.*

Sutcliffe, R. F. E., McElligott, A., O'Sullivan, D., Polikarpov, A. A., Kuzmin, L. A., Néill, G. O. & Véronis, J. (1996b). IWNR - extending a public multilingual taxonomy to russian. *Artificial Intelligence and the Simulation of Behaviour (AISB-1996) : 2nd Tutorial and Workshop on Multilinguality in the Lexicon*, 14–25.

Thorndike, E. L. (1948). On the frequency of semantic changes in modern english. *Journal of General Psychology*, *66*, 319–327.

Valli, A. & Véronis, J. (1999). Etiquetage grammatical de corpus oraux : Problèmes et perpectives. In (Vol. IV, pp. 113–133). Champs-sur-Marne: Association pour le traitement informatique des langues (ASSTRIL).

Véronis, J. (1998). A study of polysemy judgements and inter-annotator agreement. *Programme and Advanced Papers of the Senseval Workshop.*

Véronis, J. (2001). Sense tagging : Does it makes sense. *Corpus Linguistics.*

Véronis, J. (2003). Cartographie lexicale pour la recherche d'Information. *10ème conférence sur le Traitement Automatique des Langues Naturelles (TALN-2003)*, *1*, 265–274.

Véronis, J. & Ide, N. (1990). Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. *13th International Conference on Computational Linguistics (COLING-1990)*, *2*, 389–394.

Viegas, E., Mahesh, K. & Nirenburg, S. (1998). Semantics in action. *Text, Speech and Language Technology : Predicative Forms in Natural Language and in Lexical Knowledge Bases*.

Voorhees, E. M. (1993). Using WordNet to disambiguate word senses for text retrieval. *Association for Computing Machinery Special Interest Group on Information Retrieval (ACM-SIGIR-1993) : 16th Annual International Conference on Research and Development in Information Retrieval*, 171–180.

Vossen, P. (1998). Introduction to EuroWordNet. *Computers and the Humanities : Special Issue on EuroWordNet*, *32*(2–3), 73–89.

Waltz, D. & Pollack, J. (1985). Massively parallel parsing : A strongly interactive model of natural language interpretation. *Cognitive Science*, *9*, 51–74.

Weaver, W. (1955). Translation. In N. William & A. D. Booth (Eds.), *Machine translation of languages* (pp. 15–23). New-York: J. Wiley and Sons. (Reprinted from *Mimeographed*, 1949, 12 pp.)

Weiss, S. F. (1973). Learning to disambiguate. *Information Storage and Retrieval*, *9*, 33-41.

Wilks, Y. (1968). On-line semantic analysis of english texts. *Mechanical Translation*, *11*(3–4), 59–72.

Wilks, Y. (1969). Getting meaning into the machine. *New Society*, *361*, 315–317.

Wilks, Y. (1973). An artificial intelligence approach to machine translation. *Computer Models of Thought and Language*, 114–115.

Wilks, Y. (1975a). An intelligent analyzer and understander of english. *Communications of the Association for Computing Machinery (ACM)*, *18*(5), 264–274.

Wilks, Y. (1975b). Preference semantics. *Formal Semantics of Natural Language*, 329–348.

Wilks, Y. (1975c). A preferential pattern-seeking semantics for natural language inference. *Artificial Intelligence*, *6*, 53–74.

Wilks, Y. (1975d). Primitives and words. *Interdisciplinary Workshop on Theoretical Issues in Natural Language Processing*, 42–45.

Wilks, Y. (1998). Making preferences more active. *Artificial Intelligence*, *11*(3), 197–223.

Wilks, Y. & Stevenson, M. (1997a). Combining independent knowledge source for word sense disambiguation. *Conference « Recent Advances in Natural Language Processing »*, 1–7.

Wilks, Y. & Stevenson, M. (1997b). The grammar of sense : Using part-of-speech tags as first step in semantic disambiguation. *Journal of Natural Language Engineering*, *4*(3).

Wilks, Y. & Stevenson, M. (1998). Word sense disambiguation using optimised combinations of knowledge sources. *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL-1998)*, 1398–1402.

Wilks, Y. A., Fass, D. C., Guo, C. M., MacDonald, J. E., Plate, T. & Slator, B. A. (1990). *Providing machine tractable dictionary tools.* Cambridge, Massachusetts: MIT Press.

Witten, I. H. & Frank, E. (2000). *Data mining (practical machine learning tools and techniques with JAVA implementations).* Morgan Kaufmann.

Yarowsky, D. (1992). Word sense disambiguation using statistical models of roget's categories trained on large corpora. *14th International Conference on Computational Linguistics (COLING-1992)*, 454–460.

Yarowsky, D. (1993). One sense per collocation. *ARPA Workshop on Human Language Technology*, 266–271.

Yarowsky, D. (1994a). A comparision of corpus-based techniques for restoring accents in spanish and french text. *2nd Annual Workshop on Very Large Text Corpora*, 19–32.

Yarowsky, D. (1994b). Decision lists for lexical ambiguity resolution : Application to accent restoration in spanish and french. *32nd Annual Meeting of the Association for Computational Linguistics (ACL-1994)*, 88–95.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. *33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, 189–196.

Yarowsky, D. (2000). Hierarchical decision list for word sense disambiguation. In *Computers and the humanities* (Vol. 34, pp. 179–186). Netherlands: Kluwer Academic Publishers.

Yarowsky, D., Cucerzan, S., Florian, R., Schafer, C. & Wicentowski, R. (2001). The johns hopkins SENSEVAL2 system descriptions. *2nd International Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*, 163–166.

Yngve, V. H. (1955). Syntax and the problem of multiple meaning. In W. N. Locke & A. D. Booth (Eds.), *Machine translation of languages* (pp. 208–226). New York: John Wiley and Sons.

Zavrel, J., Degroeve, S., Kool, A., Daelemans, W. & Jokinen, K. (2000). Diverse classifiers for NLP disambiguation tasks comparisons, optimization, combination, and evolution. *Twente Workshops on Language Technology 18 (Learning to behave)*, 201–221.

Zernik, U. (1990). Tagging word senses in corpus : The needle in the haystack revisited. *Text-based Intelligent Systems : Current Research in Text Analysis, Information Extraction and Retrieval*(90CRD198).

Zipf, G. K. (1945). The meaning-frequency relationship of words. *Journal of General Psychology*, *33*, 251-266.